Sistemas Informáticos
Curso 2004-2005

Visualización y Animación de Estructuras de Datos y Algoritmos

Laura Gutiérrez García
Esther Rico Redondo
Carmen Torrano Giménez

Dirigido por:
Prof. Clara María Segura Díaz
Dpto. Sistemas Informáticos y Programación

Facultad de Informática
Universidad Complutense de Madrid

RESUMEN EN CASTELLANO E INGLÉS

El objetivo del proyecto es el desarrollo de una herramienta pedagógica, destinada a los alumnos universitarios pertenecientes a las facultades de informática, que estén cursando las asignaturas de estructuras de datos y esquemas algorítmicos. Esta herramienta se usará como medio didáctico para facilitar el entendimiento de dichas asignaturas.

La herramienta desarrollada es una guía visual animada para comprender el funcionamiento de estructuras de datos tales como: pilas, colas, colas de prioridad, árboles binarios de búsqueda y árboles AVL; y de diversos problemas que se pueden resolver mediante esquemas algorítmicos como divide y vencerás, voraz, ramificación y poda y programación dinámica.

The aim of the project is the development of a pedagogical tool for students of a computer science degree who are attending subjects of data structures and algorithmic schemes. This tool will be used as a didactic media in order to ease the understanding of such subjects.

The application being developed is an animated visual guide aimed to easily understand the behaviour both of data structures such as stacks, queues, priority queues, binary trees and AVL trees; and of problems than can be solved by using algorithmic schemes like divide and conquer, greedy method, branch and bound, and dynamic programming.

LISTA PALABRAS CLAVE

Acción: unidad necesaria para llevar a cabo la visualización y animación de una estructura de datos o algoritmo.

Algoritmo: conjunto de reglas que permiten obtener un resultado determinado a partir de ciertas reglas definidas.

Animación: Conjunto de acciones destinadas a visualizar de forma animada el efecto producido tras la ejecución de las mismas sobre las estructuras de datos y algoritmos.

Esquema Algorítmico: plantilla proporcionada por una metodología de programación para resolver una colección de algoritmos de forma uniforme, que resuelven problemas específicos.

Estructuras de datos: colección de datos cuya organización se caracteriza por las funciones de acceso que se usan para almacenar y acceder a elementos individuales de datos.

Simulación: secuencia de acciones que muestran el comportamiento de forma animada sin necesidad de que el usuario introduzca los datos.

Visualización: efecto producido al mostrar en pantalla las estructuras de datos y algoritmos.

ÍNDICE

1. Introducción

Este documento es una especificación de Requisitos Software (ERS) para el Sistema de Información “Visualización y Animación de Estructuras de Datos y Algoritmos”. Esta especificación se ha estructurado basándose en las directrices dadas por el estándar “*IEEE recommended Practice for Software Requirements Specification ANSI/IEEE 830 1998*”.

1.1. Propósito

El objeto de la especificación es definir de manera clara y precisa todas las funcionalidades y restricciones del sistema que se desea construir. El documento va dirigido al equipo de elaboración del presente proyecto y a los usuarios finales del sistema. Este documento será el canal de comunicación entre las partes implicadas, tomando parte en su confección miembros de cada parte.

Ésta especificación está sujeta a revisiones por el profesor y el equipo de elaboración, que se recogerán por medio de sucesivas versiones del documento, hasta alcanzar su aprobación. Una vez aprobado servirá de base para la construcción del nuevo sistema.

1.2. Alcance de la Aplicación

El objetivo del proyecto es el desarrollo de una herramienta pedagógica, destinada a los alumnos universitarios pertenecientes a las facultades de informática, que estén cursando las asignaturas de estructuras de datos y esquemas algorítmicos. Esta herramienta se usará como método didáctico para facilitar el entendimiento de dichas asignaturas.

La aplicación contará con una interfaz gráfica intuitiva sobre la que se ilustrará el funcionamiento de diversas estructuras de datos y esquemas algorítmicos.

Las estructuras de datos disponibles en la aplicación serán las siguientes:

- Pilas
- Colas
- Árboles (ABB, AVL)
- Colas de Prioridad

El usuario podrá consultar o modificar cada una de las estructuras anteriores, únicamente a través de las operaciones permitidas en las mismas. También podrá ver la especificación

algebraica de la estructura seleccionada, las diversas formas de implementación de las estructuras y el coste espacial y temporal de cada implementación.

Los algoritmos disponibles en la aplicación se ajustan a los siguientes esquemas algorítmicos:

- Programación dinámica
- Divide y vencerás
- Devorador
- Ramificación y poda

Sobre cada uno de los esquemas algorítmicos se desarrollarán algoritmos concretos, de modo que el usuario pueda ver la utilidad de cada esquema algorítmico. En particular algunas de las funcionalidades ofrecidas al usuario serán la visualización del efecto de utilizar diversas estrategias voraces, la construcción de la matriz de programación dinámica, la selección de los caminos mínimos en el algoritmo de Dijkstra, el seguimiento de las llamadas recursivas realizadas en un algoritmo de divide y vencerás...

1.3. Referencias

- *IEEE Recommended Practice for Software Requirements Specification*. ANSI/IEEE std 830, 1998.

1.4. Visión General del Documento

Este documento se compone de tres secciones, en cada una de las cuales se especifican con distinto nivel de detalle los requisitos del sistema. En esta primera sección introductoria se proporciona una visión global de la ERS.

En la segunda sección se ofrece una descripción general del sistema, sin excesivo detalle, con el fin de exponer las principales funciones que se han de llevar a cabo, los datos asociados, factores, restricciones, supuestos y dependencias que afectan al desarrollo.

Por último, en la tercera sección se define de forma detallada y exacta los requisitos que debe satisfacer el sistema, identificando e indicando cuáles son las entradas, las salidas y el proceso necesario para satisfacer cada requisito.

2. Descripción General

2.1. Descripción de la Funcionalidades del Sistema

2.1.1. Subsistemas de la Aplicación

A grandes rasgos, nuestro sistema se encargará de las siguientes tareas:

- Subsistema de Visualización y Animación de las Estructuras de Datos
  - Con vista usuario de la herramienta
  - Con vista usuario de desarrollo (representación gráfica de la implementación)
- Subsistema de Visualización y Animación de los Algoritmos
- Subsistema de Documentación
- Subsistema de Simulación
- Subsistema de Ayuda
  - Índice
  - Manual de Usuario

Ahora veremos con más detalle cómo se llevan a cabo cada uno de estos puntos.

2.1.2. Subsistema de Visualización y Animación de las Estructuras de Datos

En el subsistema de visualización y animación de las estructuras de datos se pretende representar gráficamente y con animaciones cada una de las estructuras de datos que ofrece la herramienta, de modo que el usuario pueda observar el resultado de aplicar las distintas operaciones sobre cada estructura.

El subsistema contará con el siguiente conjunto de estructuras de datos:

- Pila
- Cola
- Árbol Binario de Búsqueda
- Árbol AVL
- Cola de Prioridad

A continuación se expondrá brevemente la funcionalidad de cada una de las vistas de las que dispone:

- **Vista usuario de la herramienta**: está destinada principalmente a usuarios que no conocen la implementación de cada estructura. Esta vista representa la estructura tal y como se la imaginaría un usuario sin nociones en la materia de Estructura de Datos. Se dará la opción de visualizar el estado actual de la estructura de datos, así como el estado anterior de la misma. Esta será la vista que aparecerá por defecto en la aplicación.
- **Vista usuario de desarrollo**: es la representación gráfica de la implementación escogida de la estructura de datos.

Todas las operaciones de las estructuras de datos podrán seleccionarse directamente en la interfaz o a través del menú. En el caso de que la función no pueda aplicarse por ser parcial, debido al estado de la estructura en ese momento, ésta aparecerá deshabilitada en el menú. Sin embargo, en la interfaz las funciones siempre estarán disponibles, y en caso de que la función no pueda aplicarse, saldrá un mensaje que informe al usuario del suceso impidiendo llevar a cabo tal operación y no se producirá ningún cambio sobre la estructura de datos gráfica.

2.1.3. Subsistema de Visualización y Animación de los Esquemas Algorítmicos

En el subsistema de visualización y animación de Esquemas Algorítmicos se pretende representar gráficamente con animaciones cada uno de los algoritmos que componen la herramienta, de modo que el usuario pueda observar el proceso de ejecución del mismo. El usuario podrá visualizar uno o varios ejemplos de cada uno de los esquemas algorítmicos.

El usuario podrá observar la evolución del algoritmo de una manera animada, observando el efecto producido tras utilizar distintas estrategias voraces en el problema de la mochila implementado con la metodología voraz, visualizar las distintas acciones realizadas durante la resolución de los distintos algorítmicos, seguir paso a paso la construcción de la matriz de programación dinámica, etc.

El subsistema de gestión de algoritmos contará con el siguiente conjunto de esquemas algorítmicos:

- Voraz
- Programación Dinámica
- Divide y vencerás
- Ramificación y poda

En cada algoritmo se dispondrá de las siguientes funciones:

- **Introducir datos**: El usuario introducirá los datos de entrada con los que operará el algoritmo.
- **Iniciar**: Se iniciará la ejecución y animación del algoritmo escogido.
- **Pausar**: Interrumpe la ejecución y la animación del algoritmo hasta que vuelva a seleccionarse Ejecutar.
- **Ejecutar**: Reanuda la ejecución del algoritmo en el punto donde se quedó por última vez.
- **Parar**: Detiene y finaliza totalmente la ejecución y animación del ejemplo escogido.

- **Paso a paso**: Permite visualizar la animación de un paso de la ejecución del algoritmo.
- **Velocidad**: Esta función permitirá que se ejecute más deprisa o más despacio el algoritmo, incrementando o aminorando la misma.
- **Nuevos Datos**: Finaliza el algoritmo que se estuviera ejecutando en ese momento, limpia la pantalla y muestra la misma pantalla inicial que se visualiza al insertar los datos.

En cada una de las metodologías algorítmicas se implementará al menos un problema que refleje el comportamiento de la misma. Con el esquema algorítmico voraz, programación dinámica y ramificación y poda se implementará el problema de la mochila (versión fraccionable en el voraz y entera en los otros dos) con el objetivo de que se pueda comparar el comportamiento de los diferentes esquemas algorítmicos ante problemas del mismo ámbito. Usando el esquema algorítmico voraz también se implementará el algoritmo de Dijkstra para resolver el problema de caminos mínimos. Como ejemplos de la metodología de divide y vencerás se implementarán los algoritmos de búsqueda binaria y ordenación rápida.

A continuación se mostrará el enunciado de cada problema resuelto en nuestro sistema:

Problema de la Mochila:

Nos dan n objetos y una mochila. Para i = 1, 2,…, n, el objeto i tiene un peso positivo w_i y un valor positivo v_i. La mochila puede llevar un peso que no sobrepase W. Nuestro objetivo es llenar la mochila de tal manera que se maximice el valor de los objetos transportados, respetando la limitación de capacidad impuesta.

En el caso de la mochila fraccionable suponemos que se pueden romper los objetos en trozos más pequeños, de manera que podamos decidir llevar solamente una fracción x_i del objeto i, con $0 \leq x_i \leq 1$.

En el caso de la mochila 0-1 los objetos no son fraccionables.

Problema de los caminos mínimos:

Se considerará un grafo dirigido G = <N, A> en donde N es el conjunto de nodos de G, y A es el conjunto de aristas dirigidas. Cada arista posee una longitud no negativa. Se toma uno de los nodos como nodo origen. El problema consiste en determinar la longitud del camino mínimo que va desde el origen hasta cada uno de todos los demás nodos del grafo. Este problema se resolverá mediante el algoritmo de Dijkstra.

Problema de Búsqueda:

Sea T[1..n] un vector matriz ordenado de manera no decreciente; esto es, T[i] ≤ T[j] siempre que sea 1 ≤ i ≤ j ≤ n. Sea x un elemento. El problema consiste en buscar x en el vector T para ver si se encuentra en él. Se implementa mediante el algoritmo de búsqueda binaria.

Problema de ordenación:

Sea T[1..n] un vector de n elementos. Nuestro problema consiste en ordenar estos elementos por orden ascendente. Se implementa mediante el algoritmo de ordenación rápida *Quicksor*t.

2.1.4. Subsistema de Documentación

El subsistema de documentación ofrecerá al usuario diversa información acerca de cada una de las EDs y esquemas algorítmicos.

Para cada ED se dispondrá de:

- La **especificación algebraica** de la estructura siguiendo como referencia el libro de Ricardo Peña [7].
- El **coste**: que contendrá una tabla comparativa de los costes temporales y espaciales de las distintas operaciones de la estructura de datos, en función de la implementación utilizada en la misma.
- El **código fuente en Java** de la implementación de la estructura.
- Además se dispondrá de una **ayuda adicional** que tendrá información acerca de la estructura de datos concreta.

Para cada esquema algorítmico existirá documentación con información acerca de:

- El **coste**: que contendrá una tabla en la que podrá verse el coste del ejemplo implementado con el esquema algorítmico escogido y el coste de la implementación de ese mismo ejemplo con el resto de los esquemas algorítmicos aplicables a tal ejemplo. Esta tabla permitirá comparar los costes de los diferentes tipos de algoritmos, ya que se refieren al mismo ejemplo.
- El **código fuente en Java** del ejemplo implementado con el esquema algorítmico.
- Además se dispondrá de una **ayuda adicional** que tendrá información sobre las características y el esquema general del esquema algorítmico.

2.1.5. Subsistema de Simulación

Este subsistema se encarga de todo lo referente a las simulaciones que se le proporcionan al usuario. Las simulaciones pretenden mostrar al usuario el comportamiento

de las EDs y de los algoritmos de cada esquema algorítmico, sin necesidad de que el usuario introduzca a través de la interfaz las funcionalidades que desea realizar. Las simulaciones son una serie de ejemplos predeterminados en la aplicación, con ciertas funciones sobre las EDs y los datos de entrada de un determinado problema implementado con uno de los métodos algorítmicos.

2.1.6. Subsistema de Ayuda

Este subsistema se hará cargo de todo lo relacionado a la ayuda que se proporcionará a los usuarios.

Las funciones en que descompondremos este subsistema son:

- Índice: desde el que se accederá a toda la ayuda disponible en el sistema
- Manual de usuario: contendrá las instrucciones requeridas para un correcto funcionamiento de la aplicación.

2.2. Perfiles de Usuario

Esta herramienta va dirigida fundamentalmente a alumnos de informática que cursen las asignaturas de Estructuras de Datos y de la Información y Metodología y Tecnología de la Programación. Esta aplicación también resultará útil a los profesores que imparten dichas asignaturas.

A continuación haremos algunos comentarios acerca de estas materias.
Las directrices generales propias de Informática se encuentran en

- RD 1459/1990 para Ingeniería Informática
- RD 1460/ 1990 para Ingeniería Técnica en Informática de Gestión
- RD 1461/1990 para Ingeniería Técnica en Informática de Sistemas del 26 de Octubre de 1990 (BOE 20 Noviembre 1990).

En estos reales decretos se recoge que tanto Estructuras de Datos y de la Información (EDI) como Metodología de la Programación (MTP) son troncales en los planes de estudios de las tres titulaciones.
En Ingeniería Informática e Ingeniería Técnica en Informática de Gestión EDI consta de 12 créditos y MTP de 15. En Ingeniería Técnica en Informática de Sistemas EDI cuenta con 12 créditos y MTP con 12 créditos.

La troncalidad de EDI se manifiesta en la asignatura troncal de 2º curso llamada Estructuras de Datos y de la Información (EDI) en las tres titulaciones. Esta asignatura es de 15 créditos (10 teóricos y 5 prácticos) en Ingeniería Informática y de 12 créditos (8 teóricos y 5 prácticos) en las titulaciones técnicas.

La troncalidad de MTP se divide en las asignaturas de Programación Orientada a Objetos (POO) y Metodología de la Programación (MTP). MTP es una asignatura troncal de 3er curso. En todas las titulaciones POO cuenta con 4.5 créditos y MTP con 12 (8 teóricos y 4 prácticos). La suma de los créditos de ambas asignaturas alcanza los créditos de la troncalidad de MTP.

Los descriptores oficiales del plan de estudios de la Facultad de Informática de la Universidad Complutense de Madrid son:

EDI

- Diseño de algoritmos recursivos.
- Tipos abstractos de datos: especificación e implementación.
- Estructuras de datos y algoritmos de manipulación.
- Estructuras de la Información: ficheros, bases de datos.

MTP

- Diseño de algoritmos.
- Análisis de algoritmos.
- Lenguajes de Programación.
- Diseño de programas: descomposición modular y documentación.
- Técnicas de verificación y prueba de programas.

Pueden consultarse algunos de los programas de la asignatura de EDI impartida el curso 2004-2005 en la Facultad de Informática de la Universidad Complutense de Madrid.

Programa de EDI 2º A Ingeniería Informática
(http://www.fdi.ucm.es/profesor/csegura/edi0405/programa.doc)
Programa de EDI 2º B Ingeniería Técnica en Informática de Gestión
(http://www.fdi.ucm.es/profesor/milanjm/edi0405/EDI_0405.pdf)

Los conocimientos recomendados para las personas que usen esta herramienta son:

- temas de verificación de programas y descomposición modular.
- conocimientos básicos de un lenguaje imperativo (instrucciones, tipos de datos, procedimientos).
- diseño descendente de programas.
- lógica de primer orden (aplicación a la especificación de propiedades que debe cumplir un programa).
- realización de demostraciones por inducción.
- diseñar programas pequeños mediante refinamientos sucesivos.
- decidir qué ED elemental es más adecuado utilizar (vector, matriz, fichero...).

Anteriormente, las facultades aplicaban una política de prerrequisitos para matricularse de ciertas asignaturas. Es decir, para poder matricularse de determinadas asignaturas era

necesario aprobar alguna otra (esta asignatura es prerrequisito de la que se desea matricular) o estar matriculado de alguna otra (esta asignatura es correquisito de la que se quiere matricular).

Actualmente la política de prerrequisitos y correquisitos se ha eliminado, pero se recomienda a los alumnos que cursen algunas asignaturas prácticas impartidas en laboratorios a la vez que cursan EDI o MTP, dada la relación de tales asignaturas. Se recomienda que se curse Laboratorio de Programación II y POO simultáneamente a EDI, y Laboratorio de Programación III (en Ingeniería Informática) simultáneamente a MTP.

Debido a lo citado anteriormente, pueden darse situaciones en las que haya alumnos cursando EDI y que no hayan aprobado las asignaturas de Introducción a la Programación, Lógica o Matemática Discreta. Así mismo, puede darse la situación de alumnos que cursen simultáneamente EDI y MTP. Lo ideal es que los alumnos sigan la idea de las pautas marcadas por los prerrequisitos, ya que unos conocimientos requieren de otros anteriores que formen su base. Los pre/correquisitos dan una idea de las asignaturas cuyo contenido está relacionado.

En EDI es muy importante observar la eficiencia de los algoritmos que manipulan EDs y comparar las distintas implementaciones de las EDs y elegir la que resulte más adecuada. De la misma manera, en MTP es importante el análisis de la eficiencia para comparar distintos algoritmos que resuelven el mismo problema. Por eso se incluyen varias implementaciones de las EDs y los costes, en cada una de las implementaciones, asociados a sus operaciones. Al igual que en las EDs, en los algoritmos también se incluye el código de los mismos y el coste comparativo con otros algoritmos.

En este tipo de asignaturas es muy importante el trabajo continuado propio y la realización de múltiples ejercicios y ejemplos para poner en práctica los conocimientos adquiridos y fomentar el encuentro de soluciones a los problemas tratados por estas asignaturas.

Por eso pensamos que esta herramienta será de gran utilidad y aportará grandes beneficios a los alumnos principiantes que se inicien en las asignaturas de EDI o MTP ya que su parte gráfica y visual ayuda a aclarar y comprender los conceptos. También puede servir como motivación a los alumnos repetidores, que pueden estar más interesados en la parte de la documentación (implementación, costes...).

No olvidemos que esta herramienta también puede servir de apoyo aquellos alumnos que estudian a distancia, por libre o con autoformación.

Tras cursar la asignatura de EDI se pretende que los alumnos adquieran unos conocimientos que esperamos que nuestra herramienta les ayude a adquirir:

- Analizar la eficiencia temporal y espacial de los algoritmos.
- Razonar sobre la corrección de los algoritmos.

- Diseñar y transformar algoritmos recursivos.
- Implementar tipos abstractos de datos con estructuras de datos y determinar cómo influyen el coste de los programas.

Esta herramienta servirá de ayuda para el temario de EDI relacionado con las EDs, por lo que es recomendable que los alumnos hayan adquirido los conocimientos impartidos en EDI anteriormente (análisis de la eficiencia temporal y espacial, corrección de los algoritmos, diseño y transformación de algoritmos recursivos) .

Nuestra aplicación ayudará a adquirir y asentará los conocimientos que se adquieren tras cursar la asignatura de MTP:

- Análisis avanzado de la eficiencia de los programas.
- Transformación de algoritmos recursivos a iterativos.
- Resolver problemas con uno de los esquemas de resolución: divide y vencerás, método devorador, programación dinámica, ramificación y poda y precondicionamiento.
- Estudiar la complejidad de los programas y clasificarlos en base a ella.

Alguna de la bibliografía que usan los alumnos es:

Peña, R.; *Diseño de programas. Formalismo y abstracción;* Tercera edición. Prentice Hall, 2004.;

Kaldewaij, A.; *The Derivation of algorithms;* Prentice Hall, 1990.;

Horowitz, E.; Sahni, S.; Mehta, D.; *Fundamentals of Data Structures in C++;* W. H. Freeman & Co., 1995;

Martí Oliet, N.; Ortega Mallén, Y.; Verdejo López, J. A.; *Estructuras de datos y métodos algorítmicos:*
Ejercicios resueltos; Colección Prentice Practica, Pearson/Prentice Hall, 2003;

Franch Gutiérrez, X.; *Estructuras de Datos: Especificación, Diseño e Implementación*. Ediciones UPC, 1994.

Weiss, M. A.; *Estructuras de datos en JAVA*. Addison Wesley, 2000.

Carrano, F. M., Prichard, J. J.; *Data Abstraction and Problem Solving with Java: Walls and Mirrors*. Addison Wesley, 2001.

Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; Stein, C.; *Introduction to Algorithms. Second Edition*. The MIT Press, 2001.

2.3. Restricciones Principales

Las restricciones principales a las que se enfrenta nuestra aplicación son las siguientes:

- Restricciones de tiempo en cuanto a los plazos de entrega.
- No existen restricciones de coste, ya que las licencias las proporciona la Universidad, al igual que los equipos informáticos.
- No existen restricciones de reglas de negocio ya que por el momento no se piensa comercializar el producto.
- La aplicación debe ser amigable, siendo fácil de manejar y entender, además de proporcionar diversas pantallas de ayuda.
- El software debe ser soportado por los equipos que se adapten a las características especificadas en los requisitos no funcionales.
- Una restricción de diseño a tener en cuenta, es el espacio en disco del que dispondremos para guardar los archivos que generamos con la aplicación.
- La aplicación debe ser eficiente y el tiempo de respuesta debe ser aceptable.
- La aplicación debe seguir las normas establecidas en la especificación de requisitos, así como los criterios descritos en todos los documentos del proyecto.
- El sistema debe ser robusto.
- Todos los módulos del sistema deben mantener una estrecha relación con los demás y asegurar que toda la información contenida en ellos es coherente con la del resto de los módulos.
- Crear un software modular al que se le puedan añadir fácilmente nuevas funcionalidades.
- Restricciones de seguridad: en el momento en el que se ponga la aplicación en una página web, se deberán incluir restricciones de seguridad para que el sistema esté protegido frente a cualquier intento de acceso no autorizado al sistema.

2.4. Suposiciones y Dependencias

2.4.1. Suposiciones.

Se asume que los requisitos descritos en este documento son estables una vez aprobados por todos los componentes del equipo y el profesor de la asignatura.

Cualquier petición de cambio en la especificación de requisitos, debe ser estudiada y aprobada por todas los miembros del grupo. Además, dicha petición de cambio requerirá una mayor inversión de tiempo y recursos, pudiéndose hacer realidad los riesgos potenciales de tiempo y cambio de requisitos.

2.4.2. Dependencias.

La aplicación no contará con ninguna dependencia externa, es decir, funcionará autónomamente sin necesidad de conexión con otros sistemas externos.

2.5. Requisitos Futuros

Los requisitos futuros que pueden contemplarse son:

- Implementar nuevas estructuras de datos, tales como listas, grafos, árboles hilvanados...
- Implementar más algoritmos para cada esquema algorítmico.
- Realización de las páginas web de documentación con XML.
- Integrar la aplicación en una página web a la cual puedan acceder alumnos de cursos posteriores.
- Hacer pruebas masivas con la aplicación para estudiar su utilidad.

3. Requisitos Específicos

3.1. Requisitos Funcionales

3.1.1. Subsistema de Visualización y Animación de las Estructuras de Datos

3.1.1.1. Subsistema de Visualización y Animación de la Pila

En cada una de las vistas de la pila, podrá verse de forma animada el comportamiento de la pila al aplicar sobre ella las operaciones disponibles. Sobre la estructura de datos pila podrán aplicarse las siguientes operaciones:

- Crear
- Apilar
- Desapilar
- Consultar la cima
- Consultar si la pila está vacía
- Consultar el tamaño de la pila

A continuación veremos con detalle cada una de las operaciones disponibles para el usuario de una pila, independientemente de la vista en la que se encuentre.

- ***Crear***: Al seleccionar esta opción se creará una pila general vacía sobre la que podrán aplicarse el resto de las operaciones. Previamente se deberá indicar el tipo de los elementos que contendrá la pila. Si se selecciona esta opción

habiéndose creado una pila previamente, se empezará desde cero con una nueva estructura.

o ***Apilar***: Al apilar un elemento podrá verse de manera animada cómo se introduce un nuevo elemento en la pila. El elemento que se introduzca en la pila será el indicado por el usuario y debe ser del tipo indicado al crear la pila. Para poder realizar esta operación será necesario que la pila se haya creado con anterioridad.

El tamaño máximo del texto introducido será de 6 dígitos.

Si se introducen más elementos de los que es posible ver en la pantalla se activará la barra de desplazamiento vertical, si nos encontramos en la vista herramienta, u horizontal, si estamos en la vista de desarrollo, de modo que puedan visualizarse todos los elementos de la pila desplazando la barra.

Puede observarse que el valor del elemento aparece centrado en el nodo y que el tamaño del nodo se adapta al tamaño del texto introducido.

o ***Desapilar***: Para realizar esta operación será necesario que la pila se haya creado y que contenga algún elemento.

Cuando se desapile un elemento podrá verse de manera animada cómo se elimina el elemento de la parte superior de la pila.

Si al desapilar un elemento es posible visualizar en la pantalla todos los nodos de la pila, desaparecerá la barra de desplazamiento.

o ***Consultar la cima***: Si se selecciona esta opción, la aplicación devolverá el valor del elemento de la cima de la pila, mostrando la solución en la etiqueta descriptiva. Esta función no provoca cambios en el estado de la pila, por lo que el dibujo de la pila no sufrirá ninguna variación.

Para poder aplicar esta operación será necesario que la pila se haya creado y que contenga algún elemento.

o ***Consultar si la pila está vacía***: Si se selecciona esta opción, la aplicación indicará si la pila está vacía o si contiene algún elemento, mostrando la solución en la etiqueta descriptiva. La pila se encontrará vacía cuando se cree o cuando se hayan desapilado todos sus elementos. Esta función no provoca cambios en el estado de la pila, por lo que el dibujo de la pila no sufrirá ninguna variación.

Para poder aplicar esta operación será necesario que la pila se haya creado.

o ***Consultar el tamaño de la pila***: El tamaño de la pila indica el número de elementos que contiene la pila en ese momento. La aplicación mostrara el resultado en la etiqueta descriptiva. Esta función no provoca cambios en el estado de la pila, por lo que el dibujo de la pila no sufrirá ninguna variación.

Para poder aplicar esta operación será necesario que la pila se haya creado.

- **Con vista usuario de la herramienta**

La representación de la pila será una pila general, formada por nodos que crecen o decrecen verticalmente. En esta implementación no habrá limitaciones de tamaño de la pila. Los paneles de dibujo están colocados en vertical.

Explicaremos las características particulares, para esta vista, de algunas de las operaciones de la pila.

- ***Crear***: Al seleccionar esta opción aparecerá una pila general vacía representada como una toma de tierra. La pila es ilimitada.
- ***Apilar***: Al apilar un elemento podrá verse de manera animada cómo se introduce un nuevo elemento en la pila. El elemento se colocará en la parte superior de la pila, es decir, en la posición siguiente de la cima de la pila. La cima de la pila estará indicada por una flecha. La pila crecerá en vertical.
- ***Desapilar***: Cuando se desapile un elemento podrá verse de manera animada cómo se elimina el elemento de la parte superior de la pila, es decir, el situado en la posición de la cima de la pila, que estará indicada por una flecha.

- **Con vista usuario de desarrollo**

En la vista usuario de desarrollo de la pila, podrá verse de forma animada el comportamiento de la pila al aplicar sobre ella las operaciones disponibles. La representación de la pila dependerá de la implementación. Los paneles estarán colocados en horizontal. La pila crecerá horizontalmente. Las implementaciones posibles de la pila son:

- Implementación estática basada en arrays
- Implementación dinámica basada en listas enlazadas

Las operaciones aplicables sobre la pila para cualquiera de las implementaciones serán las mismas que las de la pila general.

Implementación estática

En el caso de que se haya seleccionado la implementación estática, la pila se mostrará con forma de vector. La capacidad del vector es de 15 elementos.

Las características particulares, para esta vista, de algunas de las operaciones de la pila son:

- ***Crear***: Al seleccionar esta opción aparecerá una pila general vacía representada como un array de 15 elementos vacío.
- ***Apilar***: Al apilar un elemento podrá verse de manera animada cómo se introduce un nuevo elemento en la pila. El elemento se colocará en la parte

derecha de la pila (al final del vector), es decir, en la siguiente posición a la cima de la pila. La cima de la pila estará indicada por una flecha.

El vector tiene una capacidad limitada, de 15 elementos. Si el tamaño de la pila es de 15 elementos y se apila un elemento, aparecerá un mensaje indicando que no es posible apilar el elemento en esta vista, debido a que la implementación con un vector hace que la pila tenga un tamaño limitado. Sin embargo, puede trabajarse con las otras vistas, en las cuales no está limitado el tamaño de la pila.

- ***Desapilar***: Cuando se desapile un elemento podrá verse de manera animada cómo se elimina el elemento de la parte derecha de la pila, es decir, el situado en la posición de la cima de la pila, que estará indicada por una flecha.

Implementación dinámica

En el caso de que se haya seleccionado la implementación dinámica, la pila se mostrará con forma de lista enlazada. Cada nodo está unido al siguiente mediante un puntero. El elemento del fondo de la pila apunta a null. En esta implementación no habrá limitaciones de tamaño de la pila.

Las características particulares, para esta vista, de algunas de las operaciones de la pila son:

- ***Crear***: Al seleccionar esta opción aparecerá una pila general vacía representada como una toma de tierra.
- ***Apilar***: Al apilar un elemento podrá verse de manera animada cómo se introduce un nuevo elemento en la pila. Al apilar se verá cómo se añade por la izquierda un nuevo nodo y el enlace de este nodo al nodo cima de la pila. El nuevo elemento introducido es ahora la cima de la pila y se indicará con una flecha.
- ***Desapilar***: Cuando se desapile un elemento podrá verse de manera animada cómo se elimina el nodo de la parte izquierda de la pila, es decir, el situado en la posición de la cima de la pila, que estará indicada por una flecha. También se eliminará su enlace con el siguiente nodo.

3.1.1.2. Subsistema de Visualización y Animación de la Cola

En cada una de las vistas de la cola, podrá verse de forma animada el comportamiento de la cola al aplicar sobre ella las operaciones disponibles. Sobre la estructura de datos cola podrán aplicarse las siguientes operaciones:

- Crear
- Añadir

- Eliminar
- Consultar el primero
- Consultar si la cola está vacía
- Consultar el tamaño de la cola

Las características de cada una de estas operaciones son, con independencia de la vista en la que trabaje el usuario, las siguientes:

- ***Crear***: Cuando se selecciona la opción crear se debe indicar el tipo de los elementos que contendrá la cola. Si se selecciona esta opción habiéndose creado una cola previamente, se empezará desde cero con una nueva estructura. Esta operación es necesaria para poder aplicar el resto de las operaciones.
- ***Añadir***: Se añadirá el elemento indicado por el usuario en la cola, el cual deberá ser del mismo tipo que el que se indicó al crear la cola. Para poder realizar esta operación será necesario que la cola se haya creado con anterioridad. El elemento introducido se colocará el último en la cola.

  Puede observarse que el valor del elemento aparece centrado en el nodo y que el tamaño del nodo se adapta al tamaño del texto introducido.
- ***Eliminar***: Se eliminará de la cola el primer elemento que figure en la misma. No se podrá eliminar en el caso de que la cola esté vacía o no se haya creado de antemano. Al eliminar el primer elemento, el siguiente elemento pasará a ser el primer elemento de la cola.
- ***Consultar el primero***: Se devolverá el valor del elemento que se sitúa primero en la cola, mostrando el resultado en la etiqueta descriptiva. Para poder aplicar esta operación será necesario que la cola se haya creado y que contenga algún elemento. Esta función no provoca cambios en el estado de la cola de prioridad, por lo que el dibujo de la cola de prioridad no sufrirá ninguna variación.
- ***Consultar el tamaño de la cola***: Se devuelve el número de elementos que contiene la cola en ese momento, mostrando la solución en la etiqueta descriptiva.

  Esta función no provoca cambios en el estado de la cola, por lo que el dibujo de la cola no sufrirá ninguna variación.

  Para poder aplicar esta operación será necesario que la cola se haya creado.
- ***Consultar si la cola está vacía***: Si se selecciona esta opción, la aplicación indicará en la etiqueta descriptiva si la cola está vacía o si contiene algún elemento. La cola se encontrará vacía cuando se cree o cuando se hayan

extraído todos sus elementos. Esta función no provoca cambios en el estado de la cola, por lo que el dibujo de la cola no sufrirá ninguna variación.

Para poder aplicar esta operación será necesario que la cola se haya creado.

Los elementos insertados aparecerán centrados en sus respectivos nodos. El tamaño de los nodos será del orden del tamaño del elemento de mayor longitud que contengan. Existirá una longitud máxima permitida a la hora de insertar un elemento, la cual será del orden de seis caracteres.

- **Con vista usuario de la herramienta**

En esta vista la representación de la cola será una cola de camiones que esperan repostar en una gasolinera.

A continuación veremos con detalle el efecto, a nivel de representación, de cada una de las operaciones disponibles, que provocan un cambio en el estado de la cola, para el usuario de una cola.

- ***Crear***: Al seleccionar esta opción se mostrará una gasolinera vacía, esto es, un surtidor en el que no hay ningún camión esperando para repostar.
- ***Añadir***: Al añadir un elemento podrá verse de manera animada cómo se incorpora al final de la cola, es decir, a la izquierda de los demás camiones que esperan ser repuestos, un nuevo camión que contiene el elemento introducido. Si no hay ningún camión en la cola, el camión insertado será el primero y último.
- ***Eliminar***: Esta operación permite extraer de la cola el primer elemento que figure en la misma, es decir hace que el camión que se sitúa más cerca del surtidor (más a la derecha en la cola) termine de repostar y se vaya.

- **Con vista usuario de desarrollo**

En la vista usuario de desarrollo de la cola podrá verse de forma animada el comportamiento de la cola a nivel de implementación al aplicar sobre ella las operaciones disponibles. La representación de la cola dependerá de la implementación seleccionada. Las implementaciones posibles de la cola son:

- Implementación estática basada en vectores circulares
- Implementación dinámica basada en listas enlazadas

Implementación estática

En el caso de que se haya seleccionado la implementación estática, la cola se mostrará como un vector circular con dos índices, primero y libre, que apuntarán a la

posición del primer elemento de la cola y a la primera posición libre, respectivamente. El comportamiento de la cola, a nivel de representación, al aplicar cada una de las operaciones disponibles, que producen cambios en el estado de la cola, deberá ajustarse a la implementación estática.

Operaciones:

- ***Crear***: Al seleccionar esta opción se mostrará una cola vacía, cuya representación será un vector circular vacío en el que los índices primero y último apuntan a la primera posición de la cola.
- ***Añadir***: Al añadir un elemento podrá verse de manera animada cómo se introduce un nuevo elemento en la cola. El elemento introducido se colocará el último en la cola, es decir, en la posición a la que apunta libre. Tras esta inserción la posición de libre se actualizará apuntando a la primera posición que se encuentre libre.
- ***Eliminar***: Esta operación permite extraer de la cola el primer elemento que figura en la misma, el que está siendo apuntado por el índice primero. Tras la eliminación, si en la cola hay más elementos, el segundo elemento que figuraba en la misma se convertirá en el primero y pasará a ser apuntado por el índice primero. En caso de que no hubiera más elementos nos encontraremos con un vector vacío en el que los índices primero y último apuntan a la misma posición, a la posición que estaba siendo apuntada anteriormente por libre.

Implementación dinámica

En el caso de que se haya seleccionado la implementación dinámica, la cola se mostrará con forma de lista enlazada. El comportamiento de la representación gráfica de cola, al aplicar las operaciones que producen cambio en su estado, deberá ajustarse a esa implementación.

Operaciones:

- ***Crear***: Al seleccionar esta opción se mostrará una cola vacía, cuya representación será el de dos punteros primero y último apuntando a null.
- ***Añadir***: Al añadir un elemento podrá verse de manera animada cómo se introduce un nuevo elemento en la cola. El elemento introducido se colocará el último en la cola, es decir, pasará a situarse en un nodo que está siendo apuntado por el elemento que con anterioridad era el último. Tras la inserción se actualizará el puntero último, de modo que apunte al elemento que acabamos de introducir.

  Si antes de la inserción la cola se encuentra vacía, el elemento introducido será primero y último en la cola: estará apuntado por los dos punteros.

El nodo que contiene al elemento introducido apuntará a null.

- ***Eliminar***: Esta operación permite extraer de la cola el primer elemento que figura en la misma, el que está siendo apuntado por el puntero primero. Tras la eliminación, si en la cola hay más elementos, el segundo elemento que figuraba en la misma se convertirá en el primero y pasará a ser apuntado por el puntero primero. En caso de que no hubiera más elementos nos encontraremos ante la cola vacía.

3.1.1.3. Subsistema de Visualización y Animación del Árbol

En cada una de las vistas del árbol, podrá verse de forma animada el comportamiento de la ED al aplicar sobre ella las operaciones disponibles. Siempre deberán cumplir la propiedad siguiente: los hijos izquierdos de cualquier nodo han de ser menores que él, mientras que los hijos derechos deberán ser mayores. Sobre la estructura de datos árbol podrán aplicarse las siguientes operaciones:

- Crear
- Insertar
- Eliminar
- Realizar un recorrido en preorden del árbol
- Realizar un recorrido en recorrido postorden del árbol
- Realizar un recorrido en recorrido inorden del árbol
- Consultar los elementos de un determinado nivel
- Consultar la altura del árbol
- Consultar el hijo izquierdo del árbol
- Consultar el hijo derecho del árbol
- Consultar la raíz del árbol
- Consultar si la estructura arbórea está vacía

Las características de cada una de estas operaciones son, con independencia de la vista en la que trabaje el usuario, las siguientes:

- ***Crear***: Cuando se selecciona esta operación se debe indicar el tipo de los elementos que contendrá la estructura arbórea. Si se selecciona esta opción habiéndose creado un árbol previamente, se empezará desde cero con una nueva estructura. Esta operación es necesaria para poder aplicar el resto de las operaciones.
- ***Insertar***: Se añadirá el elemento indicado por el usuario en la estructura arbórea, el cual deberá ser del mismo tipo que el que se indicó al crear la estructura. Para poder realizar esta operación será necesario que el árbol se haya creado con anterioridad.

El elemento se deberá insertar en el lugar adecuado, respetando la propiedad de orden existente entre ellas

o ***Eliminar***: Se eliminará de la estructura arborescente el elemento que indique el usuario. No se podrá eliminar si el elemento no se encuentre en el árbol o no se ha creado la estructura, mostrando un mensaje de error que informe del hecho.

Si la estructura arbórea sólo consta del elemento que se va a suprimir, el elemento se borrará y se mostrará el árbol vacío.

Si el árbol consta de más de un nodo, ese elemento se eliminará y el resto de los nodos se deberán reordenar animadamente, manteniendo la relación de orden.

Si el nodo a eliminar tiene hijo derecho e izquierdo, o si sólo tiene hijo derecho, colocaremos en la posición que ocupaba el elemento, el menor elemento del hijo derecho. Si sólo tiene hijo izquierdo colocaremos éste en la posición del elemento a eliminar. Para visualizar mejor el intercambio se mostrará una flecha indicando el elemento que va a sustituir al anterior, y el elemento a eliminar se mostrará tachado con un aspa.

o ***Realizar un recorrido en preorden del árbol***: Se realiza el recorrido en preorden del árbol, el cual consiste en la raíz seguida del preorden del hijo izquierdo y del preorden del hijo derecho. Para poder realizar esta operación será necesario que el árbol se haya creado con anterioridad.

o ***Realizar un recorrido en recorrido postorden del árbol***: Se realiza el recorrido en postorden del árbol, el cual consiste en la realización del postorden del hijo izquierdo seguido del postorden del hijo derecho y de la raíz. Para poder realizar esta operación será necesario que el árbol se haya creado con anterioridad.

o ***Realizar un recorrido en recorrido inorden del árbol***: Se realiza el recorrido en inorden del árbol, el cual consiste en el recorrido en inorden del hijo izquierdo, seguido de la raíz y del inorden del hijo derecho. Para poder realizar esta operación será necesario que el árbol se haya creado con anterioridad.

o ***Consultar los elementos de un determinado nivel***: Dado un determinado nivel introducido por el usuario, se devuelven todos los elementos pertenecientes a ese nivel. Los niveles empiezan en 0 y coinciden con la altura. Esta operación sólo podrá aplicarse si el árbol se haya creado y contiene algún elemento.

Se iluminarán los elementos del árbol pertenecientes al nivel introducido por el usuario.

o ***Consultar la altura del árbol***: Se devolverá la altura del árbol, mostrando la solución en la etiqueta descriptiva. Para poder aplicar esta operación será necesario que el árbol se haya creado.

- ***Consultar el hijo izquierdo del árbol***: Se devolverá el hijo izquierdo de la raíz del árbol. Si el árbol no tuviera hijo izquierdo, se informará en la etiqueta descriptiva que no existe hijo izquierdo. Para poder aplicar esta operación será necesario que el árbol se haya creado y contenga algún elemento.

  Se iluminará el hijo izquierdo de la raíz, quedando el resto del árbol sombreado.
- ***Consultar el hijo derecho del árbol***: Se devolverá el hijo derecho de la raíz del árbol. Si el árbol no tuviera hijo, se informará en la etiqueta descriptiva que no existe hijo derecho. Para poder aplicar esta operación será necesario que el árbol se haya creado y contenga algún elemento.

  Se iluminará el hijo derecho de la raíz, quedando el resto del árbol sombreado.
- ***Consultar la raíz del árbol***: Se devolverá, en la etiqueta descriptiva, el elemento que ocupa la raíz del árbol, es decir el valor del elemento situado en el primer nivel del árbol. Para poder aplicar esta operación será necesario que el árbol se haya creado y contenga algún elemento.

  Se iluminará la raíz del árbol, quedando el resto del árbol sombreado.
- ***Consultar si la estructura arbórea está vacía***: Se indicará, en la etiqueta descriptiva, si la estructura arbórea está vacía o si contiene algún elemento. El árbol se encontrará vacío cuando se cree o cuando se hayan extraído todos sus elementos.

  Para poder aplicar esta operación será necesario que el árbol se haya creado.

Todas las operaciones de consulta o recorrido no provocan cambios en el estado de la estructura arborescente, por lo que el dibujo no sufrirá ninguna variación.

Las operaciones de recorrido mostrarán de forma animada, sobre la estructura arbórea, la secuencia de elementos que componen el recorrido. Además se visualizará una cadena de texto sobre el panel de dibujo, que muestre la secuencia completa.

En todas las vistas los elementos insertados aparecerán centrados en sus respectivos nodos. El tamaño de los nodos será del orden del tamaño del elemento del árbol de mayor longitud. Existirá una longitud máxima permitida a la hora de insertar un elemento, la cual será del orden de seis caracteres.

Los tipos de árboles que se podrán visualizar son:

- Árbol binario de búsqueda
- Árbol AVL

3.1.1.3.1. Árbol Binario de Búsqueda

Los ABB permitirán la inserción de datos duplicados.

- **Con vista usuario de la herramienta**

La representación de la estructura será la de un árbol semejante a los que se dan en la naturaleza de tipo genealógico o esquemático de clasificación, formado por una raíz, que se situará en la parte inferior del panel y ramas y hojas, los cuales crecerán de abajo a arriba.

A continuación veremos con detalle características particulares de los ABB.

- ***Crear***: Se creará un árbol vacío sobre el cual el usuario podrá realizar las restantes operaciones del árbol, cuya representación será la de un tronco desnudo.
- ***Insertar elemento***: Al insertar un elemento podrá verse de manera animada cómo se introduce un nuevo elemento en la estructura arbórea creada. El elemento se añadirá en una hoja encima de las hojas construidas.

- **Con vista usuario de desarrollo**

En la vista usuario de desarrollo del ABB, podrá verse de forma animada el comportamiento de la ED a nivel de implementación al aplicar sobre ella las operaciones disponibles. La representación de árbol dependerá de la implementación seleccionada. Las implementaciones posibles del ABB son:

- Implementación estática basada en arrays o listas: el árbol se representará con un vector. Cada casilla del vector representa una posición de un árbol completo recorrido en anchura.
- Implementación dinámica basada en punteros: el árbol se representará con nodos dispuestos jerárquicamente, enlazados con punteros, de modo que cada nodo tiene dos hijos, uno a la izquierda y otro a la derecha. El árbol crecerá de arriba a abajo del panel, empezando por la raíz la cual se situará en la parte más alta del panel. La representación de las operaciones es muy semejante a la representación de la vista usuario, por lo que no haremos más hincapié en ella. Tan sólo decir que la representación del árbol binario de búsqueda vacío consistirá en una toma de tierra.

Implementación estática

En el caso de que se haya seleccionado la implementación estática, el árbol se mostrará con forma de array de modo que los nodos rellenos se encuentren en

color y los no rellenos en blanco y negro. El comportamiento del árbol al aplicar las operaciones sobre ella deberá ajustarse a esa implementación.

- ***Crear***: Se creará un árbol vacío sobre el cual el usuario podrá realizar las restantes operaciones del árbol, cuya representación será una toma de tierra.
- ***Insertar elemento***: Al insertar un elemento podrá verse de manera animada cómo se introduce un nuevo elemento en el vector que representa al árbol. El hijo izquierdo de un elemento dado se encontrará en la posición 2^n del vector y el elemento derecho en la 2^{n+1}. Al insertar pueden quedar huecos no rellenos en el vector, esto es debido a que el árbol es no completo.
- ***Eliminar elemento***: El usuario podrá eliminar un nodo cualquiera del árbol. La eliminación contribuirá a que queden más huecos en el vector.

3.1.1.3.2. Árbol AVL

Se dispondrá, al igual que en el resto de las estructuras, de dos vistas (vista usuario de desarrollo y vista usuario de la herramienta), pero a diferencia de las otras estructuras en la vista usuario de desarrollo careceremos de implementación estática, no tiene sentido implementar un árbol AVL con un vector.

Los AVL son árboles equilibrados, de modo que para todos los nodos del árbol las alturas de los subárboles derecho e izquierdo deben diferir a lo sumo en uno.

La representación será equivalente a la usada con los ABB, por tanto sólo explicaremos las diferencias existentes entre los AVL con respecto a los ABB.

- ***Inserción***

  Tras la inserción del nuevo elemento, el árbol debe estar equilibrado,. Si tras la inserción el árbol se desequilibra habrá que reequilibrarlo. Hay dos tipos de rotaciones para equilibrar un árbol: rotación a la izquierda o a la derecha, dependiendo de cual sea el subárbol que presente mayor altura. Estas rotaciones podrán ser simples o compuestas. Cuando haya que equilibrar se mostrará el subárbol a equilibrar en diferente color al resto del árbol, señalando el tipo de rotación (izquierda o derecha) mediante una flecha. Para visualizar mejor la rotación se hará por pasos de modo que se muestren los cambios de punteros que se producen a lo largo de la rotación. Cada vez que se produzca un cambio de posición de un nodo, se indicará el puntero que hemos destruido y la unión que se va a realizar mediante una flecha que indicará dónde se va a colocar.
- ***Eliminación***

  El elemento indicado se eliminará de forma que el árbol no se desequilibre. Por tanto cuando se elimine estaremos realizando una

eliminación equivalente a la que hacíamos en los árboles binarios seguida de un equilibrado (si es necesario) que se efectuará de la misma forma que en la inserción de los árboles AVL.

A diferencia de la inserción, a la hora de realizar una eliminación pueden producirse varios desequilibrios.

3.1.1.4. Subsistema de Visualización y Animación de la Cola de Prioridad

En cada una de las vistas de la cola de prioridad, podrá verse de forma animada el comportamiento de la ED al aplicar sobre ella las operaciones disponibles. La cola de prioridad desarrollada es un montículo de mínimos. Sobre la ED podrán aplicarse las siguientes operaciones:

- Crear
- Insertar
- Eliminar el elemento mínimo
- Consultar el elemento mínimo
- Consultar si la cola de prioridad está vacía
- Consultar el tamaño de la cola de prioridad

Las características de cada una de estas operaciones son, con independencia de la vista en la que trabaje el usuario, las siguientes:

- ***Crear***: Se deberá indicar el tipo de los elementos que contendrá la cola de prioridad. Si se selecciona esta opción habiéndose creado una cola de prioridad previamente, se empezará desde cero con una nueva estructura. Esta operación es necesaria para poder aplicar el resto de las operaciones.
- ***Insertar***: Se añadirá el elemento indicado por el usuario en la cola de prioridad, el cual deberá ser del mismo tipo que el que se indicó al crear la estructura. Para poder realizar esta operación será necesario que la cola de prioridad se haya creado con anterioridad.

  El elemento se insertará siempre en la última posición de la cola, para posteriormente ir flotando hasta la posición que le corresponda.

  Puede observarse que el valor del elemento aparece centrado en el nodo y que el tamaño del nodo se adapta al tamaño del texto introducido.
- ***Eliminar mínimo***: Se eliminará el elemento mínimo de la estructura, el cual es el que ocupa la primera posición de la misma. Este elemento será sustituido por el elemento que ocupa la última posición de la cola, el cual deberá hundirse posteriormente.

  Para realizar esta operación será necesario que la cola de prioridad se haya creado y que contenga algún elemento.

- ***Consultar el elemento mínimo***: Se devolverá el valor del elemento con mayor prioridad, mostrando el resultado en la etiqueta descriptiva. Para poder aplicar esta operación será necesario que la cola de prioridad se haya creado y que contenga algún elemento. Esta función no provoca cambios en el estado de la cola de prioridad, por lo que el dibujo de la cola de prioridad no sufrirá ninguna variación.
- ***Consultar si la cola de prioridad está vacía***: Se indicará en la etiqueta descriptiva si la cola de prioridad está vacía o si contiene algún elemento. La cola de prioridad se encontrará vacío cuando se cree o cuando se hayan extraído todos sus elementos. Esta función no provoca cambios en el estado de la estructura, por lo que el dibujo no sufrirá ninguna variación.

  Para poder aplicar esta operación será necesario que la cola de prioridad se haya creado.
- ***Consultar el tamaño de la cola de prioridad***: Se devolverá el número de elementos que contiene la cola de prioridad en ese momento, mostrando la solución en la etiqueta descriptiva. Esta función no provoca cambios en el estado de la cola de prioridad, por lo que el dibujo de la cola de prioridad no sufrirá ninguna variación.

  Para poder aplicar esta operación será necesario que la cola de prioridad se haya creado.

El valor del elemento aparecerá centrado en un nodo, y el tamaño de tal nodo se adaptará al tamaño del texto de mayor longitud introducido. El tamaño máximo del texto introducido estará limitado a seis caracteres.

- **Con vista usuario de la herramienta**

En la vista usuario de la cola de prioridad podrá verse cómo se van añadiendo uno a uno los elementos introducidos por el usuario y cómo se va actualizando el valor del elemento mínimo. La representación de la cola de prioridad será la de un conjunto de cajas ordenadas según han sido añadidas por el usuario, sobre las cuales se señalizará el elemento mínimo con una flecha.

A continuación veremos con detalle la representación de cada una de las operaciones disponibles para el usuario de una cola de prioridad, que provocan cambios en el estado de la cola de prioridad.

- ***Crear***: Al seleccionar esta opción aparecerá una cola de prioridad vacía sobre la que podrán aplicarse el resto de las operaciones. La representación de la cola vacía será la de una toma de tierra.

- ***Insertar***: Se añadirá el elemento introducido por el usuario al final de los nodos previamente insertados. Si el elemento insertado posee más prioridad que el resto, se actualizará la flecha que señala el mínimo.
- ***Eliminar mínimo***: Se eliminará el elemento de mayor prioridad actualizándose el nuevo mínimo.

- **Con vista usuario de desarrollo**

Se dispondrá de dos representaciones en la vista usuario de desarrollo:

- Representación de la implementación estática del montículo de mínimos
- Visualización arbórea del montículo

Implementación estática del montículo de mínimos

Se visualizará el montículo como un vector ordenado de mayor a menor prioridad.

- ***Crear***: Al seleccionar esta opción aparecerá una cola de prioridad vacía sobre la que podrán aplicarse el resto de las operaciones. La representación de la cola vacía será la de una toma de tierra.
- ***Insertar***: Se añadirá el elemento introducido por el usuario en la posición que le corresponda según su prioridad. Para ello primero se hará sitio al elemento, mostrando los intercambios necesarios mediante flechas, y posteriormente se introducirá el elemento animadamente.
- ***Eliminar mínimo***: Se eliminará el elemento de mayor prioridad desplazándose para ello los elementos que sea necesario. Posteriormente se actualizará el mínimo, mostrando los intercambios entre elementos mediante flechas.

Visualización arbórea del montículo

Se representará la cola de prioridad mediante un montículo, estructura arbórea en la cual el elemento de la raíz será menor o igual que el resto de los elementos del árbol. Además, los subárboles izquierdo y derecho también deberán deben ser montículos. El árbol representado será un árbol casi completo.

Operaciones:

- ***Crear***: Al seleccionar esta opción aparecerá un montículo vacío sobre el que podrán aplicarse el resto de las operaciones. La representación del montículo vacío será la de una toma de tierra.
- ***Insertar***: Tras insertar un elemento en el montículo, se podrá ver de forma animada cómo se sitúa en la posición adecuada, de manera que no se pierda la

relación de orden. Si al realizarse la inserción deja de ser un montículo, se deberá flotar el elemento hacia la raíz, esto es, intercambiar el elemento con su padre, hasta que se cumpla la propiedad del montículo. Para representar la inserción primero haremos sitio al elemento, mostrando los intercambios necesarios mediante flechas, y posteriormente introduciremos el elemento animadamente.

o ***Eliminar mínimo***: En la eliminación del elemento mínimo se observará de manera animada cómo se elimina el elemento de mayor prioridad (el elemento que ocupa la raíz del montículo) disminuyendo el tamaño de la estructura de datos. Para llevar a cabo la eliminación del elemento mínimo se colocará el último elemento en la raíz y se aplicará la función de hundir hasta que se cumpla la propiedad del montículo. Esta operación consistirá en intercambiar el elemento con su hijo más pequeño. Los intercambios entre elementos se mostrarán a través de flechas.

3.1.2. Subsistema Documentación de las Estructuras de Datos

Para cada estructura de datos se dispondrá de la siguiente documentación:

- ***Especificación algebraica***: El usuario contará con la funcionalidad de contemplar la especificación algebraica de la estructura de datos concreta. La especificación algebraica de la estructura es independiente de la implementación y muestra el conjunto de operaciones que es posible realizar con ella.
- ***Código***: Se dispondrá de la posibilidad de acceder al código en Java con el que se ha implementado la estructura. Se mostrará el código correspondiente a la implementación escogida en la vista usuario de desarrollo. En el caso de que se haya escogido en la vista usuario de la herramienta, se mostrará el código de todas las implementaciones posibles.
- ***Coste***: Otra de las posibilidades es visualizar una tabla que permita comparar el coste asintótico temporal y espacial de cada una de las operaciones de la estructura de datos en las diferentes implementaciones.
- ***Ayuda adicional***: Pulsando esta opción se dispondrá de información general sobre la estructura de datos.

3.1.3. Subsistema de Visualización y Animación de los Esquemas Algorítmicos

Para llevar a cabo la visualización y animación de los esquemas algorítmicos, el usuario deberá seleccionar el problema que desea ejecutar e introducir los datos de partida que son necesarios para la resolución del mismo. Una vez seleccionado el problema a resolver e introducido los datos requeridos, se deberá seleccionar la opción de iniciar para comenzar la visualización y animación del problema en cuestión.

Tras pulsar la opción de iniciar se podrá observar la ejecución del problema. Si se desea interrumpir la ejecución se deberá seleccionar la opción de Pausar, la reanudación se producirá tras seleccionar la opción de Ejecutar. La finalización de la ejecución se llevará a cabo con la opción Parar.

Otra de las opciones puestas a disposición, consiste en la ejecución paso a paso del algoritmo, de manera que al pulsar esa opción se podrá observar un paso de la ejecución del problema. En todo momento será posible realizar la ejecución completa desde el punto donde se quedo, seleccionando la opción de Ejecutar comentada más arriba.

Además, se podrá variar la velocidad de ejecución del problema seleccionando la opción Velocidad. Así como comenzar desde cero pulsando la opción Nuevos Datos.

En los esquemas algorítmicos con más de un problema, se podrá cambiar de aplicación en cualquier momento y el efecto producido será el mismo que Nuevos Datos.

3.1.3.1. Subsistema de Visualización y Animación del Esquema Algorítmico Voraz

Como ejemplos se implementarán el algoritmo voraz que resuelve el problema de la mochila fraccionable y el algoritmo de Dijkstra. Abordaremos estos problemas con una estrategia voraz a través de una serie de etapas. En cada etapa tomaremos una decisión y no reconsideramos esta decisión.

Problema de la mochila fraccionable

Los datos de partida solicitados para la ejecución del problema de la mochila serán el número de objetos que podrán introducirse en la mochila, así como el peso y valor de cada uno de ellos, la capacidad máxima de la mochila y la estrategia voraz con la que se desea llevar a cabo la resolución del problema. Al iniciar la ejecución se mostrará la relación valor/peso existente entre cada objeto.

En todo momento se podrá ver gráficamente la mochila construida hasta el momento con los objetos insertados en ella. Se visualizará la construcción de la mochila paso a paso, insertando los objetos seleccionados en cada etapa hasta alcanzar el tamaño máximo de la mochila. Además se visualizará una tabla, que va mostrando en cada etapa del proceso voraz, el valor y peso almacenado en la mochila, el candidato seleccionado y las porciones de cada objeto insertadas en ella. También se mostrará los datos de salida, esto es, las porciones finales de cada objeto que forman parte de la mochila.

Algoritmo de Dijkstra

Los datos de partida solicitados para la ejecución del algoritmo de Dijkstra son el número de nodos que forman parte del grafo y el peso de las aristas.

En este caso se visualizará un grafo cuyos nodos representen las ciudades del recorrido y cuyas aristas determinen la distancia existente entre las mismas.

Durante la ejecución del algoritmo se mostrará en cada etapa, con distintos colores, los nodos procesados hasta el momento y el nodo seleccionado (nodo cuya distancia al nodo origen es mínima). También se visualizará los nodos, no procesados hasta el momento, cuya distancia al nodo origen se vea modificada por el nuevo nodo seleccionado.

Como datos de salida se mostrará sobre el grafo en un color más oscuro, los nodos y las aristas que formen parte de su camino mejor, y en una tabla se reflejará la distancia mínima que hay entre cualquier nodo y el nodo inicio.

3.1.3.2. Subsistema de Visualización y Animación del Esquema Algorítmico de Programación Dinámica

Como ejemplo se implementará el algoritmo que resuelve el problema de la mochila 0-1.

Problema de la mochila 0-1

Los datos de partida solicitados para la ejecución del problema de la mochila serán el número de objetos que podrán introducirse en la mochila, así como el peso y valor de cada uno de ellos, y la capacidad máxima de la mochila.

Al resolver el problema de la mochila podrá visualizarse la matriz con los objetos en las filas y el peso de la mochila en las columnas (desde cero hasta el peso máximo de la mochila). La matriz se irá rellenando con el valor de la mejor mochila, cuyo peso se corresponde con el indicado en la columna, utilizando los primeros i elementos, siendo i la fila. Cuando se haya rellenado toda la tabla se tendrá la solución de la mejor mochila con peso correspondiente a la capacidad máxima y utilizando todos los objetos, en la última celda de la matriz.

Una vez construida toda la matriz, se reconstruirá la solución, mostrando los objetos que se han seleccionado en la mochila, el beneficio y el tamaño ocupado de la mochila, como datos de salida.

Todos los pasos realizados durante la construcción de la matriz, así como la reconstrucción de la solución, se podrán visualizar paso a paso coloreando las celdas implicadas en la operación en cuestión de la matriz solución y del vector de entrada.

3.1.3.3. Subsistema de Visualización y Animación del Esquema Algorítmico Divide y Vencerás

En este subsistema se desarrollarán dos problemas típicos de la metodología divide y vencerás: el algoritmo de búsqueda binaria y el algoritmo Quicksort.

Algoritmo Búsqueda Binaria

Este problema consiste en obtener la posición en la que se encuentra un elemento dentro de un vector ordenado. En el caso de que el elemento no se encuentre en el vector, se devolverá la posición en donde debería situarse el elemento si se hallara en el vector. La solución se basa en buscar el elemento en el punto medio del vector, si el elemento no se encuentra en dicha posición se divide en dos mitades el vector y se busca el elemento en el subvector izquierdo si el elemento buscado es menor que el elemento del punto medio, o en el subvector derecho si es mayor.

El usuario introducirá los datos de partida utilizados para la resolución del problema. Estos datos estarán formados por el número de elementos del vector a tratar, el elemento a buscar, así como los elementos sobre los que se va a realizar la búsqueda, los cuales no tienen que esta ordenados ya que el algoritmo inicialmente realizará una ordenación.

En el panel de dibujo podrá verse el vector ordenado, sobre el cual se buscará la posición de un determinado elemento, mediante sucesivas divisiones de la parte que se procesa del vector.

La parte que se está procesando del vector aparecerá en color mientras que la otra parte aparecerá en blanco y negro.

Inicialmente la parte procesada será todo el vector. Se comprobará si el elemento a buscar se haya en la parte central de la parte procesada del vector. Si es así se habrá encontrado la solución, sino se comparará el elemento del medio con el elemento buscado, en función de lo cual se seleccionará la mitad donde seguir buscando. Si es

más pequeño se cogerá como parte a procesar la mitad izquierda, sino se seleccionará la mitad derecha. Esto se hará hasta que no queden más elementos por procesar o bien hasta que se encuentre la solución.

Todo este método de resolución será mostrado de forma animada en una serie de pasos. En cada paso se mostrará el vector junto con las comparaciones que se estén llevando a cabo.

Algoritmo Quicksort

El problema consiste en ordenar los elementos de un vector. El método de resolución de este problema se basa en la idea de dividir el vector por la mitad e ir ordenando los subvectores.

Inicialmente se le pedirá al usuario los datos de partida, los cuales estará compuestos por el número de elementos que compondrán el vector a ordenar y el valor de los elementos a ordenar. Una vez introducidos los datos de partida se mostrará el vector de entrada.

En los sucesivos pasos de la animación se seleccionará un elemento pivote (elemento situado en la primera posición del vector), se colocará los elementos más grandes que el elemento pivote a la derecha y los más pequeños a su izquierda. Una vez que se tenga esta semi-ordenación se procesará por separado, de la misma manera que la descrita anteriormente, la parte situada a la derecha del pivote y la parte izquierda. Este procesamiento se irá repitiendo hasta que cada parte del vector quede ordenada. Una vez que todas las partes estén ordenadas se tendrá el vector completo ordenado.

3.1.3.4. Subsistema de Visualización y Animación del Esquema Algorítmico de Ramificación y Poda

Como ejemplo se implementará el algoritmo que resuelve el problema de la mochila 0-1.

Problema de la mochila 0-1

Los datos de partida solicitados para la ejecución del problema de la mochila serán el número de objetos que podrán introducirse en la mochila, capacidad de la mochila, peso y valor de cada uno de los objetos.

Al ejecutarse el algoritmo, podrá visualizarse el árbol a partir del cual se busca la solución y la cola de prioridad asociada a él. La cola y el árbol se corresponden unívocamente. Como el problema es de maximización, se utilizará una cola de prioridad de máximos, cuyos elementos serán nodos del mismo tipo que los nodos del árbol. En los nodos podrá visualizarse la información del nodo correspondiente a la etapa, peso, beneficio y beneficio óptimo.

El comportamiento del árbol es el de un árbol genérico, pero en el problema de la mochila basta con que sea un árbol binario. El comportamiento de la cola de prioridad es el mismo que el de la estructura de datos explicada anteriormente, teniendo en cuenta que se trata de una cola de prioridad de máximos.

Al desarrollarse el algoritmo podrán visualizarse los nodos del árbol que se generan y la evolución de la cola asociada. Los nodos de los que se vayan a generar los hijos se insertarán en la cola. Al sacar un nodo de la cola se generarán sus nodos hijos. Cuando la cota de un nodo sea peor que la mejor solución encontrada hasta el momento no se generaran sus nodos hijos, por lo cual no se insertará dicho nodo en la cola. Al no insertar los nodos en la cola, se poda. Cuando la cola quede vacía se habrá finalizado el recorrido del árbol.

En el problema de la mochila, cada nivel del árbol representará uno de los objetos. Cada arista representará la decisión de si se inserta o no el objeto en la mochila. Cada nodo representa un llenado parcial de la mochila. Sólo se dibujarán los nodos del árbol (y de la cola) que no se hayan podado.

También se mostrará el resultado del problema indicando qué objetos se han seleccionado para introducirse en la mochila y cuales no.

3.1.4. Subsistema de Documentación de Algoritmos

Para cada problema de los esquemas algorítmicos se dispondrá de la siguiente documentación:

- Código: Se tendrá la opción de visualizar el código con el que se ha implementado el ejemplo en el esquema algorítmico concreto.
- Coste: Podrá verse una tabla que contendrá el coste temporal y espacial del ejemplo implementado con el esquema algorítmico seleccionado, así como el coste de la implementación de este mismo ejemplo con el resto de los esquemas algorítmicos aplicables al problema.

- Ayuda Adicional: Se dispondrá de la información referente a las características y el esquema general del esquema algorítmico.

3.1.5. Subsistema de Simulación

Para facilitar la tarea al usuario, se le proporcionarán una serie de simulaciones, con las cuales se podrá disponer de una serie de ejemplos concretos, tanto de las EDs como de los algoritmos, con los que podrá estudiar el comportamiento de cada uno de ellos. Se podrá trabajar fácilmente con la aplicación sin necesidad de que el usuario tenga que indicar a través de la interfaz las operaciones de las estructuras de datos que desea realizar o insertar los datos del problema algorítmico.

En el caso de las estructuras de datos, se contará con un fichero que tendrá las operaciones que se desean ejercer sobre la ED. La aplicación leerá las operaciones de la ED y mostrará de forma animada el comportamiento desencadenado por tales operaciones. Esta opción se habilitará tan sólo en la vista usuario de la herramienta, salvo en la ED Cola de prioridad que se habilitará en la visualización arbórea. Esta opción sólo se ha desarrollado en una de las vistas por falta de tiempo, quedará como trabajo futuro extender esta opción en todas las vistas.

Con respecto a los algoritmos, la aplicación almacenará los datos del ejemplo concreto. Por defecto se iniciará la ejecución del algoritmo, en cualquier momento podrá seleccionarse el resto de las opciones propias de los esquemas algorítmicos.

3.1.6. Subsistema de Ayuda

Desde el índice se podrá acceder a toda la ayuda disponible en el sistema. Éste poseerá hipervínculos a los documentos de cada estructura de datos y esquema algorítmico, explicados más arriba.

El manual de usuario contendrá el soporte técnico necesario para el correcto funcionamiento del sistema. En él se describirá las posibles acciones que se pueden llevar a cabo, así como los resultados esperados que se han de obtener.

3.2. Requisitos de Interfaces Externas

3.2.1. Interfaces de Usuario

La interfaz estará compuesta por varias ventanas y deberá encuadrarse en el entorno de Windows. Ha de ser completa, intuitiva y fácil de manejar, y debe contener todos los componentes necesarios para realizar las funciones disponibles en la aplicación.

Las ventanas de las que constará la interfaz son las siguientes: una ventana de bienvenida, una ventana de selección y la ventana principal.

3.2.1.1. Ventana de Bienvenida

Esta ventana introductoria dará la bienvenida a los usuarios que hagan uso de la aplicación y le mostrará en breves palabras el objetivo y la funcionalidad de la misma.

3.2.1.2. Ventana de Selección

Desde esta ventana se dará opción de visualizar y animar una estructura de datos o un esquema algorítmico. Una vez seleccionado lo que se desea visualizar y animar, se mostrará las posibles estructuras o los posibles algoritmos concretos con los que se podrá trabajar. Tras escoger uno de ellos, se pasará a la ventana principal de la ED o esquema algorítmico seleccionado.

3.2.1.3. Ventana Principal de las Estructuras de Datos

La ventana principal de las EDs estará compuesta por los componentes siguientes:

- Un menú desplegable.
- Una etiqueta que describa la acción realizada sobre la ED.
- Un panel gráfico para representar el estado actual.
- Un panel gráfico para representar el estado anterior.
- Una caja con todas las funciones asociadas a una ED. Desde esta caja se podrá aplicar cualquiera de las funciones, de manera que si se intenta ejecutar alguna función no aplicable en ese momento, se mostrará un mensaje de error que informe sobre el problema y el motivo del fallo.
- Un panel con todas las acciones llevadas a cabo desde el momento en que se creo la ED.

Desde el menú desplegable de la ventana principal de las EDs se puede acceder a todas las funciones permitidas en la aplicación.

El menú desplegable está estructurado como se muestra a continuación:

- Menú Herramientas
  - Submenú Visualizar: desde el que se puede moldear la interfaz al gusto del usuario. Se podrá visualizar, si así lo desea, la descripción de la acción, el panel gráfico que contiene el estado actual, el panel gráfico que contiene el estado anterior y el panel de funciones compuesto por la caja de

acciones que se pueden llevar a cabo sobre la ED, y la lista de acciones realizadas sobre la misma.

- o Submenú Funciones: contiene las acciones que se ponen a disposición para trabajar con la estructura de datos. Las acciones no factibles aparecerán deshabilitadas, y por el contrario las acciones permitidas estarán activas para su uso.
- o Submenú Volver a Inicio: cierra la ventana principal de la ED y vuelve a la ventana de Selección.
- o Submenú Salir: cierra la aplicación.

- Menú Vista
  - o Submenú Usuario de la herramienta: visualiza la ED concreta con independencia de la implementación. Esta vista irá destinada a usuarios sin conocimiento previo de las EDs, de forma que se familiarice con ellas a partir de ejemplos cotidianos de la vida. Por ejemplo en la cola se mostrará una cola de camiones que esperan a ser repuestos en la gasolinera.
  - o Submenú Usuario de desarrollo: visualiza la ED concreta con relación a una de sus posibles implementaciones. Por ejemplo, la pila se podrá representar con una implementación estática (en forma de vector) o con una implementación dinámica (en forma de lista enlazada).
- Menú Documentación
  - o Submenú Especificación algebraica: muestra la especificación algebraica correspondiente a la ED con el que se esté trabajando.
  - o Submenú Código: muestra el código fuente en Java correspondiente a la ED con la que se esté trabajando.
  - o Submenú Costes: muestra una tabla que represente los distintos costes espaciales y temporales, en función de la implementación utilizada en la ED.
  - o Submenú Adicional: muestra información adicional acerca de la ED con la que se esté trabajando.
- Menú Ayuda
  - o Submenú Índice: contiene un catálogo con todos los temas de ayuda disponibles, para facilitar el entendimiento y posibilitar la consulta, de cualquiera de las estructuras de datos y esquemas algorítmicos utilizados en la aplicación.
  - o Submenú Manual de Usuario: contiene la ayuda técnica necesaria para el correcto uso de la aplicación.

3.2.1.4. Ventana Principal de los Esquemas Algorítmicos

La ventana principal de los Esquemas Algorítmicos estará compuesta por los componentes siguientes:

- Un menú desplegable.
- Una etiqueta que describe los pasos a seguir y las acciones realizadas sobre el problema concreto.

Desde el menú desplegable de la ventana principal de los esquemas algorítmicos, se puede acceder a todas las funciones permitidas en la aplicación. Los submenús Coste y Código del menú Documentación y el menú Simulación, aparecerán deshabilitados hasta seleccionarse el problema sobre el que trabajar.

El menú desplegable está estructurado como se muestra a continuación:

- Menú Herramientas
  - Submenú Funciones: contiene las acciones que se ponen a disposición para trabajar con el problema concreto. Inicialmente todas las acciones aparecerán deshabilitadas. Las acciones *Iniciar* y *Paso a paso* aparecerán deshabilitadas hasta seleccionarse el problema e introducir los datos de entrada necesarios para su ejecución, mientras que las acciones *Parar* y *Velocidad* aparecerán deshabilitadas hasta iniciar la resolución del problema y cuando finalice la ejecución del mismo.
  - Submenú Volver a Inicio: cierra la ventana principal del esquema algorítmico y vuelve a la ventana de Selección.
  - Submenú Salir: cierra la aplicación.
- Menú Aplicación: compuesto por los problemas concretos que se resolverán siguiendo la metodología algorítmica seleccionada.
- Menú Documentación
  - Submenú Código: muestra el código fuente en Java correspondiente al problema seleccionado, mediante el esquema algorítmico con el que se esté trabajando.
  - Submenú Costes: muestra una tabla que represente los distintos costes espaciales y temporales, en comparación con el resto de esquemas algorítmicos que resuelven el problema seleccionado, o bien en comparación con otros problemas que tienen el mismo objetivo.
  - Submenú Adicional: muestra información adicional acerca del esquema algorítmico con el que se esté trabajando o bien del problema a resolver.
- Menú Ayuda
  - Submenú Índice: contiene un catálogo con todos los temas de ayuda disponibles, para facilitar el entendimiento y posibilitar la consulta, de

cualquiera de las estructuras de datos y esquemas algorítmicos utilizados en la aplicación.

o Submenú Manual de Usuario: contiene la ayuda técnica necesaria para el correcto uso de la aplicación.

Tras seleccionar el problema a ejecutar, la ventana podrá contendrá también los siguientes componentes:

- Un panel gráfico para representar el estado actual del problema en cuestión.
- Una serie de botones bajo el panel gráfico o el panel de inserción de datos, que permiten realizar las funcionalidades propias de los esquemas algorítmicos. Dichos botones se etiquetarán como *Iniciar, Paso a paso, Parar y (- , +) Velocidad.* Los botones *Iniciar* y *Paso a paso* aparecerán siempre habilitados, de manera que si se intenta ejecutar alguna función no aplicable en ese momento, se mostrará un mensaje de error que informe sobre el problema y el motivo del fallo. Los botones *Parar* y *Velocidad* sólo aparecerán activos durante la resolución del problema.
- Un panel para la inserción de los datos del problema. En dicho panel también se podrá visualizar el resultado del mismo, y contará con un botón *Nuevos Datos* que sirva para vaciar el panel de dibujo e insertar nuevos datos.

3.2.2. Interfaces Hardware

No se han definido interfaces Hardware con sistemas externos.

3.2.3. Interfaces Software

Tampoco han sido definidas interfaces Software con otros sistemas.

3.3. Requisitos no Funcionales

3.3.1. Requisitos de Rendimiento

Nuestro software utilizará la implementación de estructuras de datos y los esquemas algorítmicos más eficientes para reducir, en todo lo posible, el tiempo de respuesta del programa.

A los datos de las simulaciones de las EDs, almacenados en ficheros, se accederá de manera rápida y eficaz.

3.3.2. Requisitos de Seguridad

Nuestro sistema no requerirá características especiales de seguridad. Todos los usuarios tendrán acceso a la aplicación y dispondrán de las mismas posibilidades de manejo de la misma.

3.3.3. Requisitos Hardware

Las características técnicas recomendadas para los equipos en lo que se ejecutará nuestra aplicación son: procesador superior a 450 MHz, memoria superior o igual 256Mb, disco duro superior a 200MB resolución de pantalla de al menos 1024 por 768 píxeles y módem o router para descargarse la aplicación.

3.3.4. Requisitos Software

Una máquina que soporte un sistema operativo Windows y un explorador de Internet. La máquina deberá disponer de conexión a Internet para descargarse la aplicación en un futuro.

3.3.5. Requisitos de Datos

3.3.5.1. Datos de Entrada

Los elementos que maneja la aplicación como entrada son los datos introducidos por el usuario, tales como la selección de visualización y animación, la estructura de datos o algoritmo concreto, los datos que se requieren al trabajar con la estructura de datos o algoritmo, el tipo de datos específico que se va a manejar en la ED...

Cuando se cree una ED, se deberá seleccionar de una lista el tipo de datos de los elementos que compondrá la estructura. Los elementos que introduzca el usuario deberán ser del tipo de datos especificado anteriormente. En caso de que no sea así, se mostrará un mensaje de error y se informará del hecho. También se pondrá una restricción en cuanto al tamaño del valor del elemento a introducir, así como del número de elementos que formarán parte de la misma en la implementación estática (por ejemplo no se podrá introducir más de 15 elementos en la pila o cola estática, ni tampoco se podrá introducir más de 4 niveles en el árbol binario o equilibrado AVL estático).

Dependiendo de cada problema concreto se requerirá distintos datos, concretados en el subsistema de visualización y animación de los esquemas algorítmicos. En todos los problemas los tipos de los datos que se manejará son enteros, en caso de introducir

datos que no sean de dicho tipo se mostrará un mensaje de error que informe del hecho.

3.3.5.2. Datos de Salida

Las salidas, de las que constará el sistema, son el resultado gráfico de aplicar las acciones sobre las EDs o problemas implementados con los esquemas algorítmicos y la documentación disponible. Si se trata de ejecutar una acción no permitida se mostrará un mensaje de error.

En el caso de las EDs se mostrará también, en la vista usuario de la herramienta, una lista con las acciones ejecutadas desde su creación. Mientras que en los esquemas algorítmicos se mostrará el resultado obtenido en el problema en cuestión, también se podrá visualizar información adicional que explique, por ejemplo, el proceso voraz seguido.

3.3.6. Requisitos de Desarrollo

El modelo de ciclo de vida que se seguirá para la elaboración del presente producto será el IEEE 1074. Este modelo de ciclo de vida nos servirá de referencia a lo largo del desarrollo del proyecto incorporando procesos para el control de la calidad del producto, procesos para el análisis, gestión y supervisión de los posibles riesgos que puedan acaecer a lo largo del desarrollo de producto, así como otros muchos que serán aplicados para conseguir un software de calidad que siga las especificaciones marcadas por el profesor de la asignatura.

Como modelo de desarrollo se seguirá el modelo en espiral de Boehm. Este modelo nos permitirá refinar el producto de acuerdo a las expectativas marcadas mediante sucesivos refinamientos (ciclos) a través de los cuales el profesor podrá ver la evolución del proyecto mediante revisiones llevadas a cabo al finalizar los ciclos. Además este modelo es muy ventajoso ya que incorpora objetivos de calidad y gestión de riesgos, los cuales permitirán eliminar alternativas no atractivas al comienzo del desarrollo.

4. Glosario

4.1. Definiciones

- Estructura de datos: Es una colección de datos cuya organización se caracteriza por las funciones de acceso que se usan para almacenar y acceder a elementos individuales de datos. Las estructuras de datos pueden descomponerse en los elementos que la forman. La manera en que se colocan los elementos dentro de la estructura afectará la forma en que se realicen los

accesos a cada elemento. La colocación de los elementos y la manera en que se accede a ellos puede ser encapsulada.

- Pila: La pila es una estructura de datos lineal homogénea, es decir, que en ella se pueden almacenar elementos de cualquier tipo, pero todos los elementos de la pila deben ser de ese mismo tipo. La pila es una estructura LIFO (*last-in*, *first-out*) ya que el último elemento insertado en la pila será el primero en salir de ella. El último elemento introducido en la pila se denomina cima.
- Cola: La cola es una estructura de datos lineal homogénea en la que los datos entran por un extremo y salen por el otro. La cola es una estructura FIFO (*first-in*, *first-out*) ya que el primer elemento de la cola será el primero en salir de ella.
- Árbol: Un árbol es una estructura de datos consistente en una colección de nodos que puede estar vacía o no. Si no está vacía, el árbol estará formado por un nodo raíz y cero o más subárboles que están unidos a la raíz por otras tantas aristas. La aplicación no contendrá esta estructura de datos.
- Árbol binario: Es un conjunto de elementos del mismo tipo tal que o bien es el conjunto vacío, en cuyo caso se denomina árbol vacío, o bien no es vacío, en cuyo caso existe un elemento distinguido llamado raíz, y el resto de los elementos se distribuyen en dos subconjuntos disjuntos, cada uno de los cuales es un árbol binario, llamados respectivamente subárboles izquierdo y derecho del árbol original.
- Árbol binario de búsqueda: Es un árbol binario ordenado en el que el valor contenido en todos los nodos internos es mayor o igual que los valores contenidos en su hijo izquierdo o en cualquiera de los descendientes de ese hijo, y menor o igual que los valores contenidos en su hijo derecho o en cualquiera de los descendientes de ese hijo.
- Árbol AVL: Es un árbol binario de búsqueda equilibrado. Para que un árbol esté equilibrado es necesario que para todos los nodos del árbol, las alturas de los subárboles derecho e izquierdo se diferencien a lo sumo en uno.
- Cola de prioridad: Es una cola cuyo primer elemento en salir es el de menor valor (en caso de una cola de prioridad de mínimos) o el de mayor valor (en el caso de una cola de prioridad de máximos).
- Montículo: Es una estructura de datos arbórea que se usa como implementación de la cola de prioridad. La estructura arbórea consiste en un árbol binario casi completo cuyos nodos incluye un elemento de información denominado valor del nodo, y que tiene la propiedad consistente en que el valor de cada nodo interno es menor (mayor) o igual que los valores de sus hijos.
- Esquema algorítmico: Son las diferentes estrategias de programación para implementar los algoritmos que resuelven los problemas. Los esquemas algorítmicos que abarca esta aplicación son: programación voraz, programación dinámica, divide y vencerás, y ramificación y poda.
- Algoritmo: Es la implementación de un ejemplo de problema concreto que se ha implementado siguiendo uno de los esquemas algorítmicos. Los algoritmos que se mostrarán

como ejemplos para visualizar los esquemas algorítmicos serán: el algoritmo que implementa el problema de la mochila, el algoritmo de búsqueda binaria, el algoritmo Quicksort y el algoritmo de Dijkstra.

4.2. Acrónimos

- EDs: Estructuras de datos
- ED: Estructura de datos
- GUI: Interfaz gráfica de usuario
- EVS: Estudio de Viabilidad del Sistema
- ABB: Árbol binario de búsqueda.
- AVL: Son las iniciales de Adelson-Velskii y Landis, los cuales idearon este tipo de árbol.

CASOS DE USO

1. Comienzo de la Aplicación

CASO DE USO #1	**Mostrar documento de bienvenida**
Objetivo en contexto	Visualización de una pantalla de bienvenida al inicio de la aplicación
Entradas	No hay
Precondiciones	El usuario arranca la aplicación
Salidas	Visualización de una pantalla de bienvenida al inicio de la aplicación en la que se informará sobre las capacidades de la herramienta.
Poscondición si éxito	No hay
Poscondición si fallo	No hay
Actores	Usuario de la herramienta o usuario de desarrollo.

Secuencia normal		
Paso	**Acción**	**Respuesta**
1	Se arranca la aplicación.	Se carga la página web "Visualización y Animación de Estructuras de Datos y Esquemas algorítmicos.htm"

CASO DE USO #2	**Iniciar aplicación**
Objetivo en contexto	Seleccionar si se desea trabajar con EDs o con algoritmos. Seleccionar el tipo de ED deseada o el tipo de esquema algorítmico deseado. La ventana que permite hacer estas selecciones estará disponible en el momento en el que se arranque la aplicación, cuando se pulse la opción de la pantalla principal "Volver a inicio" y cuando se cierre la ventana principal de la ED o esquema algorítmico.
Entradas	La selección del usuario: si desea trabajar con EDs o con algoritmos y el tipo de ED o de esquema algorítmico deseado.
Precondiciones	El usuario se encuentra en esta parte la aplicación. (Se ha pulsado el botón "Comenzar" de la pantalla de bienvenida).
Salidas	Se muestra una ventana en la que se puede seleccionar si se desea trabajar con EDs o con algoritmos. En el caso de que se haya elegido ED,

	se mostrará una lista desplegable con los tipos de EDs para que el usuario pueda seleccionar la que desee. En el caso de que se haya elegido algoritmos, se mostrará una lista desplegable con los tipos de esquemas algorítmicos para que el usuario pueda seleccionar la que desee. Además, se mostrará como salida la ventana principal de la ED o esquema algorítmico seleccionado.
Poscondición si éxito	No hay.
Poscondición si fallo	No hay.
Actores	Usuario.

Secuencia normal		
Paso	**Acción**	**Respuesta**
1.	Pulsar el botón de “Comenzar” de la ventana de bienvenida.	Mostrar una ventana en la que se pueda elegir si se desea trabajar con EDs o con algoritmos.
2.	Seleccionar si se desea trabajar con EDs o con algoritmos.	Se visualiza el combo box.
3a.	Si se seleccionó la opción de EDs	En el combo box se muestra los tipos de estructuras de datos disponibles para trabajar.
3b.	Si se seleccionó la opción de algoritmos.	En el combo box se muestra los esquemas algorítmicos disponibles para trabajar.
3.	Seleccionar el tipo de ED o esquema algorítmico concreto con el que se desea trabajar.	Mostrar la interfaz de la aplicación correspondiente al tipo de ED elegida o al tipo de esquema algorítmico seleccionado.

2. Finalización de la Aplicación

CASO DE USO #3	**Volver a Inicio**
Objetivo en contexto	Cerrar la ventana principal y volver a la ventana de selección.
Entradas	No hay.
Precondiciones	El usuario se encuentra en la ventana principal.
Salidas	Se cierra la ventana principal donde se encontraba el usuario y se muestra la ventana de selección.
Poscondición si éxito	No hay.

Poscondición si fallo	No hay.
Actores	Usuario de la herramienta o usuario de desarrollo.

Secuencia normal		
Paso	**Acción**	**Respuesta**
1.	Seleccionar la opción de "Volver a Inicio" desde el menú de herramientas.	Se cierra la ventana principal y se muestra la ventana de selección.

CASO DE USO #4	**Salir de la aplicación**
Objetivo en contexto	Salir de la aplicación
Entradas	No hay
Precondiciones	El usuario se encuentra en una de las ventanas de la aplicación
Salidas	Se cierra la ventana de la aplicación donde se encontrara el usuario
Poscondición si éxito	Se sale de la aplicación
Poscondición si fallo	No hay
Actores	Usuario de la herramienta o usuario de desarrollo.

Secuencia normal		
Paso	**Acción**	**Respuesta**
1	Seleccionar la opción de "Salir" desde el menú de herramientas.	Se sale de la aplicación

3. Configuración de la Aplicación

CASO DE USO #5	**Visualizar elementos de la interfaz**
Objetivo en contexto	Seleccionar si se desea tener visible o no en la interfaz una serie de elementos: descripción de la acción, estado actual, estado anterior y funciones.
Entradas	Selección de si se desea visualizar o no cada uno de los elementos.
Precondiciones	El usuario se encuentra en la ventana principal correspondiente a la ED o esquema algorítmico concreto.
Salidas	Activación o desactivación de la visualización seleccionada por el usuario de los elementos de la interfaz.

Poscondición si éxito	No hay.
Poscondición si fallo	No hay.
Actores	Usuario de la herramienta o usuario de desarrollo.

Secuencia normal		
Paso	**Acción**	**Respuesta**
1.	Seleccionar en el menú de herramientas la opción de "Visualizar" y el elemento de la interfaz que se desee visualizar mediante un visto. No poner un visto a los elementos de la interfaz que no se deseen visualizar.	En la ventana principal aparecerán visibles los elementos seleccionados por el usuario mediante un visto. Aquellos elementos que aparezcan sin el visto no se visualizarán.

4. Simulación

CASO DE USO #6	**Ver simulación**
Objetivo en contexto	Visualizar de forma animada el comportamiento de las EDs y los algoritmos ante unos ejemplos concretos dados, sin tener que introducir los datos desde la interfaz para observar dicho comportamiento. En los algoritmos se pueden realizar las funciones de parar, pausar, paso a paso... durante la simulación. Se contará con un fichero que contenga las instrucciones de la ED o bien se dispondrá en la aplicación de los datos de entrada del algoritmo.
Entradas	No hay.
Precondiciones	El usuario se encuentra en la pantalla principal de una ED o de un problema concreto que sigue la metodología de un esquema algorítmico.
Salidas	Animación del comportamiento de la ED o del algoritmo de acuerdo con el ejemplo de simulación elegido.
Poscondición si éxito	Se realizan las operaciones del fichero sobre la EDs o se ejecuta el algoritmo con los datos de entrada almacenados en la aplicación.
Poscondición si fallo	No hay.
Actores	Usuario de la herramienta o usuario de desarrollo.

Secuencia normal		
Paso	**Acción**	**Respuesta**

1.	Seleccionar la opción de "Simulación" del menú, en la vista donde se encuentre habilitada, y elegir el ejemplo de simulación que se desea ejecutar.	
2a.	Si el usuario se encuentra en una parte de la aplicación correspondiente a las EDs	Mostrar la visualización del comportamiento de la ED concreta en la que se encuentre el usuario. Sobre la EDs se realizarán las operaciones indicadas en el fichero. Consultar los casos de uso correspondientes a dichas operaciones.
2b.	Si el usuario se encuentra en una parte de la aplicación correspondiente a los algoritmos.	Mostrar el comportamiento del ejemplo escogido de manera animada. Los datos de entrada para el ejemplo serán los que se encuentren en la aplicación.

5. Visualización y Animación de Estructuras de Datos

5.1. Visualización y Animación de la Pila

CASO DE USO #7	**Crear pila**
Objetivo en contexto	Crear una nueva pila del tipo de la vista e implementación escogida, para poder realizar operaciones sobre la pila.
Entradas	Tipo de la pila que se desea crear.
Precondiciones	El usuario se encuentra en la parte de la aplicación de la estructura de datos pila.
Salidas	Si no se cumple alguna de las condiciones de la precondición, se muestra un mensaje de error, si no, se dibuja la pila vacía en el panel. Etiqueta que informa de la función seleccionada y del tipo de los elementos de la pila seleccionado. En el caso de que la pila ya se hubiera creado con anterioridad, se mostrará un mensaje para confirmar si se quiere comenzar de nuevo.
Poscondición si éxito	Se creará una nueva pila sobre la que se pueden realizar todas las operaciones.
Poscondición si fallo	Se mostrará un mensaje informando del motivo del fallo.

Actores	Usuario de la herramienta o usuario de desarrollo.	
Secuencia normal		
Paso	**Acción**	**Respuesta**
1.	Seleccionar la operación crear y pulsar el botón “Aplicar”.	Mostrar una ventana que pide el tipo del que se desea que se cree la pila.
2.	Seleccionar el tipo del que se desea que se cree la pila.	Comprobar en qué vista e implementación se encuentra el usuario.
3a.	Si el usuario se encuentra en la vista de usuario de la herramienta.	Se crea una nueva pila con una implementación general del tipo seleccionado. Si la pila se había creado con anterioridad se descarta la pila ya creada y se comienza con una nueva pila. Se pinta la pila vacía en el panel representada como una toma de tierra y se muestra una etiqueta que informa de la función seleccionada.
3b.	Si el usuario se encuentra en la vista de usuario de desarrollo con la implementación estática.	Se crea una nueva pila estática del tipo seleccionado. Si la pila se había creado con anterioridad se descarta la pila ya creada y se comienza con una nueva pila. Se pinta la pila vacía en el panel representada como un array de 15 elementos vacío y se muestra una etiqueta que informa de la función seleccionada.
3c.	Si el usuario se encuentra en la vista de usuario con la implementación dinámica.	Se crea una nueva pila dinámica del tipo seleccionado. Si la pila se había creado con anterioridad se descarta la pila ya creada y se comienza con una nueva pila. Se pinta la pila vacía en el panel representada como una toma de tierra y se muestra una etiqueta que informa de la opción seleccionada.

CASO DE USO #8	**Apilar**
Objetivo en contexto	Insertar un nuevo elemento en la pila.
Entradas	Elemento que se desea apilar.
Precondiciones	La pila ha sido creada, la pila no está llena, el elemento a apilar es del mismo tipo que la pila y el elemento no supera la capacidad máxima de anchura de la pila.
Salidas	Si no se cumple alguna de las precondiciones se mostrará un mensaje de error. En otro caso: animación del nuevo elemento hasta apilarse en la cima de la pila. Etiqueta que informa de la función seleccionada.
Poscondición si éxito	Se insertará en la cima de la pila el nuevo elemento.
Poscondición si fallo	Se mostrará un mensaje de error informando de los motivos por los que no es posible llevar a cabo la operación.
Actores	Usuario de la herramienta o usuario de desarrollo.

Secuencia normal

Paso	**Acción**	**Respuesta**
1.	Introducir en la etiqueta el valor que se desee apilar en la pila, seleccionar la operación apilar y pulsar el botón “Aplicar”. No es posible introducir un dato con más de 6 dígitos, por lo que no se superará el ancho permitido de la pila.	Comprobar que la pila ha sido creada.
2a.	Si la pila no se ha creado con anterioridad.	S1.
2b.	Si la pila ha sido creada.	Comprobar que el elemento es del mismo tipo que el tipo con el que fue creada la pila.
3a.	Si el elemento no es del mismo tipo que la pila.	S3.
3b.	Si el elemento es del mismo tipo que la pila.	Comprobar en qué vista e implementación se encuentra el usuario.

4a.	Si el usuario se encuentra en la vista de usuario de la herramienta.	La pila tendrá un aspecto de pila general, en posición vertical. La pila será ilimitada. Se creará un nodo para el nuevo elemento y se insertará en la cima de la pila. Podrá verse de forma animada cómo se inserta el nuevo elemento en la cima de la pila, por encima del resto de los elementos. El elemento aparecerá centrado en el nodo. Se mostrará una etiqueta informando de la función escogida. Si se introducen más elementos de los que es posible ver en la pantalla se activará la barra de desplazamiento vertical de modo que puedan visualizarse todos los elementos de la pila desplazando la barra. Si la pila contiene más de 15 elementos y se cambia a la vista estática, S4.
4b.	Si el usuario se encuentra en la vista de usuario de desarrollo con la implementación estática.	La pila tendrá aspecto de array, en posición horizontal y su capacidad máxima es de 15 elementos. Se comprueba si la pila está llena.
4b1.	Si la pila tiene 15 elementos (está llena).	S2.
4b2.	Si la pila tiene menos de 15 elementos (no está llena).	El nuevo elemento se insertará en la siguiente posición del array correspondiente a la cima de la pila (último elemento válido que rellena la pila). Podrá verse de forma animada cómo se inserta el nuevo elemento detrás del resto de los elementos de la pila. El elemento aparecerá centrado en la pila. Se mostrará una etiqueta informando de la función escogida. Si se introducen más elementos de los que es posible ver en la pantalla se activará la barra de desplazamiento vertical de modo que puedan visualizarse todos los elementos de la pila desplazando la barra.

4c.	Si el usuario se encuentra en la vista de usuario con la implementación dinámica.	La pila tendrá un aspecto de lista enlazada, en posición horizontal. La pila será ilimitada. Se creará un nodo para el nuevo elemento, se enlazará con el elemento de la cima de la pila y el nuevo elemento será la nueva cima. Podrá verse de forma animada cómo se inserta el nuevo elemento en la cima de la pila, por la izquierda. El elemento aparecerá centrado en la pila. Se mostrará una etiqueta informando de la función escogida. Si se introducen más elementos de los que es posible ver en la pantalla se activará la barra de desplazamiento vertical de modo que puedan visualizarse todos los elementos de la pila desplazando la barra. Si la pila contiene más de 15 elementos y se cambia a la vista estática, S4.
Secuencia Alternativa		
Paso	**Acción**	**Respuesta**
S1.	Pila no creada.	Mostrar un mensaje que indique que no puede ejecutarse la operación debido a que la pila no se ha creado.
S2	Pila llena.	Mostrar un mensaje en la etiqueta superior indicando que no puede apilarse debido a que la pila está llena.
S3	Tipo del elemento incorrecto.	Mostrar un mensaje indicando que el tipo del elemento es incorrecto. Se puede volver a insertar un dato para apilarlo Ir a 1.
S4	Vista no disponible.	Se mostrará un mensaje que indique que no es posible trabajar con la vista estática cuando la pila contiene más de 15 elementos.

CASO DE USO #9	**Desapilar**
Objetivo en contexto	Eliminar el elemento de la cima de la pila, es decir, el último que se insertó.

Entradas	Ninguna.
Precondiciones	La pila ha sido creada, la pila contiene algún elemento.
Salidas	Si no se cumplen las precondiciones: mensaje que informe del suceso. En otro caso: animación del elemento de la cima al desapilarse. Etiqueta que informa de la opción seleccionada.
Poscondición si éxito	La pila contendrá un elemento menos, ya que se eliminará el elemento de la cima de la pila.
Poscondición si fallo	Se mostrará un mensaje informando del motivo por el que no puede realizarse la operación.
Actores	Usuario de la herramienta o usuario de desarrollo.

Secuencia normal

Paso	**Acción**	**Respuesta**
1.	Seleccionar la operación desapilar y pulsar el botón “Aplicar”.	Comprobar que la pila ha sido creada.
2a.	Si la pila no se ha creado con anterioridad.	S1.
2b.	Si la pila ha sido creada.	Comprobar que la pila no está vacía.
3a.	Si la pila está vacía.	S2.
3b.	Si la pila no está vacía.	Comprobar en qué vista e implementación se encuentra el usuario.
4a.	Si el usuario se encuentra en la vista de usuario de la herramienta.	La pila tendrá el aspecto de una pila general, en vertical. Se eliminará el último elemento de la pila. Se verá de forma animada cómo se elimina el elemento de la cima de la pila, que se encontrará por encima de todos los demás. Se mostrará una etiqueta informando de la función escogida. Si al desapilar un elemento es posible visualizar en la pantalla todos los nodos de la pila, desaparecerá la barra de desplazamiento.

4b.	Si el usuario se encuentra en la vista de usuario de desarrollo con la implementación estática.	La pila tendrá un aspecto de array, en horizontal. Se eliminará el último elemento de la pila. Se verá de forma animada cómo se elimina el elemento de la cima de la pila, que se encontrará a la derecha de todos los demás. Se mostrará una etiqueta informando de la función escogida. Si al desapilar un elemento es posible visualizar en la pantalla todos los nodos de la pila, desaparecerá la barra de desplazamiento.
4c.	Si el usuario se encuentra en la vista de usuario con la implementación dinámica.	La pila tendrá un aspecto de lista enlazada, en horizontal. Se eliminará el último elemento de la pila. Se verá de forma animada cómo se elimina el elemento de la cima de la pila, que se encontrará a la izquierda de todos los demás. Además, se eliminará el enlace del nodo cima con el siguiente nodo. Se mostrará una etiqueta informando de la operación escogida. Si al desapilar un elemento es posible visualizar en la pantalla todos los nodos de la pila, desaparecerá la barra de desplazamiento.
Secuencia Alternativa		
Paso	**Acción**	**Respuesta**
S1.	Pila no creada.	Mostrar un mensaje que indique que no puede ejecutarse la operación debido a que la pila no se ha creado.
S2	Pila vacía.	Mostrar un mensaje indicando que no puede desapilarse debido a que la pila está vacía.

CASO DE USO #10	**Consultar la cima de la pila**
Objetivo en contexto	Consultar el valor del elemento que se encuentra en la cima de la pila.
Entradas	No hay.

Precondiciones	La pila ha sido creada, la pila no está vacía.
Salidas	Valor del elemento de la cima de la pila. El dibujo de la pila será el mismo que el de antes de aplicar esta operación. Si no se cumplen las precondiciones: mensaje de error.
Poscondición si éxito	La pila no sufrirá ninguna variación.
Poscondición si fallo	Se mostrará un mensaje informando del motivo por el que no se puede efectuar la operación.
Actores	Usuario de la herramienta o usuario de desarrollo.

Secuencia normal

Paso	**Acción**	**Respuesta**
1.	Seleccionar la operación cima y pulsar el botón “Aplicar”.	Comprobar que la pila ha sido creada.
2a.	Si la pila no se ha creado con anterioridad.	S1.
2b.	Si la pila ha sido creada.	Comprobar que la pila no está vacía.
3a.	Si la pila está vacía.	S2.
3b.	Si la pila no está vacía.	En cualquiera de las vistas: mostrar en la etiqueta superior el valor del elemento de la cima de la pila. La pila no sufrirá ninguna variación y el dibujo de la pila y las barras de desplazamiento tampoco.

Secuencia Alternativa

Paso	**Acción**	**Respuesta**
S1.	Pila no creada.	Mostrar un mensaje que indique que no puede ejecutarse la operación debido a que la pila no se ha creado.
S2	Pila vacía.	Mostrar un mensaje indicando que no puede consultarse la cima debido a que la pila está vacía.

CASO DE USO #11	**Consultar si la pila está vacía**
Objetivo en contexto	Consultar si la pila contiene algún elemento o no. La pila estará vacía cuando esté recién creada o cuando se hayan desapilado todos sus elementos.
Entradas	No hay.
Precondiciones	La pila ha sido creada.
Salidas	Etiqueta que muestre si la pila está vacía o si contiene algún elemento. El dibujo de la pila será el mismo que el de antes de aplicar esta operación.
Poscondición si éxito	La pila no sufrirá ninguna variación.
Poscondición si fallo	Mensaje que indique por qué no puede efectuarse la operación.
Actores	Usuario de la herramienta o usuario de desarrollo.

Secuencia normal

Paso	**Acción**	**Respuesta**
1.	Seleccionar la operación ¿está vacía? y pulsar el botón “Aplicar”.	Comprobar que la pila ha sido creada.
2a.	Si la pila no se ha creado con anterioridad.	S1.
2b.	Si la pila ha sido creada.	En cualquiera de las vistas: mostrar una etiqueta que informe de si la pila contiene algún elemento o no. La pila no sufrirá ninguna variación y el dibujo de la pila y las barras de desplazamiento tampoco.

Secuencia Alternativa

Paso	**Acción**	**Respuesta**
S1.	Pila no creada.	Mostrar un mensaje que indique que no puede ejecutarse la operación debido a que la pila no se ha creado.

CASO DE USO #12	**Consultar el tamaño de la pila**

Objetivo en contexto	Consultar el número de elementos que contiene la pila en ese momento. Este número se refiriere al número de elementos que se han apilado en la pila menos los que se han desapilado. Este número estará comprendido entre 0 (la pila está vacía) y el número máximo de elementos que pueda contener la pila.
Entradas	No hay.
Precondiciones	La pila ha sido creada.
Salidas	Número de elementos que contiene la pila actualmente. El dibujo de la pila será el mismo que el de antes de aplicar esta operación.
Poscondición si éxito	La pila no sufrirá ninguna variación.
Poscondición si fallo	Mensaje que indique el motivo por el que no puede realizarse la operación.
Actores	Usuario de la herramienta o usuario de desarrollo.

Secuencia normal

Paso	**Acción**	**Respuesta**
1.	Seleccionar la operación tamaño y pulsar el botón “Aplicar”.	Comprobar que la pila ha sido creada.
2a.	Si la pila no se ha creado con anterioridad.	S1.
2b.	Si la pila ha sido creada.	En cualquiera de las vistas: mostrar en la etiqueta el número de elementos que contiene la pila. La pila no sufrirá ninguna variación y el dibujo de la pila tampoco.

Secuencia Alternativa

Paso	**Acción**	**Respuesta**
S1.	Pila no creada.	Mostrar un mensaje que indique que no puede ejecutarse la operación debido a que la pila no se ha creado.

5.2. Visualización y Animación de la Cola

CASO DE USO #13	**Crear Cola**
Objetivo en contexto	Crear una estructura de datos cola
Entradas	Tipo de la estructura que se desea crear.
Precondiciones	El usuario se encuentra en la parte de la aplicación de la estructura de datos cola.
Salidas	Si no se cumple la precondición se muestra un mensaje de error, si no, se dibuja una toma de tierra que representa la cola vacía. Etiqueta que informa de la función seleccionada y del tipo de los elementos de la estructura.
Poscondición si éxito	Se crea una cola nueva.
Poscondición si fallo	Se muestra un mensaje de error, que informa de la razón del fallo.
Actores	Usuario de la herramienta o usuario de desarrollo.

Secuencia normal		
Paso	**Acción**	**Respuesta**
1.	Seleccionar la opción de "Crear" desde el menú de herramientas o desde la caja de funciones y pulsando al botón aplicar.	Mostrar una ventana que pida el tipo de los elementos que contendrá la cola.
2.	Seleccionar el tipo de los elementos de la cola.	Se crea una nueva estructura de datos interna cola, sobre la cual se realizarán el resto de las operaciones de la cola. La implementación interna usada para la cola será un array circular. Si ya existía una cola creada con anterioridad, se descarta, y se comienza con una nueva cola. Se comprueba en qué vista e implementación se encuentra el usuario.

3a.	Si el usuario se encuentra en la vista de usuario de la herramienta.	Se pinta la cola vacía en el panel y se muestra una etiqueta que informa de que la cola ha sido creada. La representación de la cola vacía en esta vista será la de una gasolinera vacía, esto es, un surtidor en el que no hay ningún camión esperando para repostar.
3b.	Si el usuario se encuentra en la vista de usuario de desarrollo con la implementación estática.	Se pinta una cola vacía en el panel cuya representación será un vector circular vacío en el que los índices primero y último apuntan a la primera posición de la cola. La forma de representar el vector circular será un array de 15 nodos donde el último elemento apuntará al primero. Se muestra una etiqueta que informa sobre la creación de la cola.
3c.	Si el usuario se encuentra en la vista de usuario con la implementación dinámica.	Se pinta la cola vacía en el panel, cuya representación será cuya representación será dos punteros primero y último apuntando a null. Se muestra una etiqueta que informa acerca de la creación de la cola.

CASO DE USO #14	**Añadir elemento a cola**
Objetivo en contexto	Añadir un nuevo elemento a la cola creada con anterioridad
Entradas	Elemento a añadir en la estructura
Precondiciones	La cola ha sido creada, la cola no está llena y el elemento es del tipo especificado al crearla
Salidas	Visualización de la cola con un nuevo nodo al final y animación de la inserción del elemento en la cola. Si no se cumple alguna de las precondiciones se mostrará un mensaje de error.
Poscondición si éxito	Se inserta el nuevo elemento al final de la cola
Poscondición si fallo	La cola permanece en el mismo estado en que se hallaba cuando se pulsó esta opción, es decir, el elemento no se inserta

Actores	Usuario de la herramienta o usuario de desarrollo.	
Secuencia normal		
Paso	**Acción**	**Respuesta**
1	Seleccionar la opción de "Añadir" desde el menú de herramientas o desde la caja de funciones y pulsando al botón aplicar. Insertar el elemento en el cuadro de texto activado.	Se comprueba que la cola se ha creado.
1.a	Si la cola se ha creado	Comprobar que el elemento introducido es del tipo especificado al crear la cola, ir a 1.c
1.b	Si la cola no se ha creado	Ir a S1
1.c	Si el elemento es del tipo especificado	Si nos encontramos en la vista usuario de desarrollo y se ha elegido la implementación estática, comprobar que la cola no supere los 15 elementos, ir a 1.e Si nos encontramos en cualquiera de las otras vistas, se añade el elemento a la cola interna de manera que aumentará su tamaño en 1, ir a 2
1.d	Si el elemento no es del tipo especificado	Ir a S2
1.e	Si nos encontramos en la vista estática y el número de elementos insertados es menor que 15.	Se añade el elemento a la cola interna de manera que aumentará su tamaño en 1, ir a 2
1.f	Si nos encontramos en la vista estática y el número de elementos insertados es 15.	Ir a S3

2a.	Si la vista seleccionada es la de usuario de la herramienta	Podrá verse de manera animada cómo se introduce el nuevo elemento en la cola en la última posición, es decir, a la izquierda de los demás camiones que esperan ser repuestos, un nuevo camión que contiene el elemento introducido. Si sólo hay un elemento en la cola, éste será el primero y último. Cada elemento se representará en el centro del remolque del camión. El tamaño de los nodos de la cola se adaptará al tamaño del texto introducido. Si al insertar un nuevo elemento no cabe en el panel, aparecerán unas barras de desplazamiento de modo que se pueda visualizar la cola en toda su dimensión.

2b	Si la vista elegida es la de usuario de desarrollo y se ha elegido la implementación estática.	El elemento se insertará de manera animada por el extremo izquierdo del array que representa la cola. El elemento introducido se colocará el último en la cola, es decir, en la posición a la que apunta libre. Tras esta inserción, la posición de libre se actualizará apuntando a la primera posición que se encuentre libre. Si sólo hay un elemento en la cola, éste será el primero y último. Debido a la representación en forma de array circular, si se ha rellenado la última casilla del array situada en la parte derecha del panel y se ha de insertar más elementos y el vector no está lleno, el siguiente elemento será el que ocupa la primera posición del array (a la izquierda del panel). Cada elemento se representará en una casilla del array en el que el valor del elemento aparecerá centrado. El tamaño de las casillas de la cola se adaptará al tamaño del texto introducido. Si debido al tamaño de los elementos que contiene la cola esta no cabe en el panel, aparecerán una barras de desplazamiento horizontal de modo que permita la visualización de la cola en toda su plenitud.

2c.	Si la vista elegida es usuario de desarrollo y se ha elegido la implementación dinámica.	El elemento se añadirá de manera animada en un nuevo nodo de la lista enlazada en la posición última de la cola, es decir se situará tras el nodo que anteriormente estaba siendo apuntado por último, de modo que este nodo le apunte. Se actualiza el puntero a último de forma que apunte al elemento que acabamos de insertar. El nodo insertado deberá apuntar a null ya que no hay ningún elemento tras él. Si la cola estaba vacía antes de la inserción, el nodo insertado además de ser último también será primero debiendo ser apuntado por el puntero primero. Cada elemento se representará en un nodo de modo que el valor del elemento aparezca centrado. El tamaño de las casillas de la cola se adaptará al tamaño del texto introducido. Los nodos se introducidos se situarán uno detrás de otro horizontalmente. Si los elementos introducidos no caben en la cola aparecerá un barra horizontal en la parte inferior del panel para permitir la visualización de la cola en toda su amplitud.
Secuencia alternativa		
Paso	**Acción**	**Respuesta**
S1	Cola no creada	Se muestra un mensaje de error que indique que la cola no se ha creado.
S2	El tipo del elemento que se desea insertar no se corresponde con el tipo de la estructura.	Si muestra un mensaje de error informando sobre la incompatibilidad de tipos y no se añade el elemento introducido.

S3	Cola llena	Se muestra un mensaje que indique que no se puede llevar a cabo la operación debido a que se han superado los límites de la cola. Si se pasa a otra vista y se intenta volver a la vista usuario de desarrollo en la implementación estática, aparecerá un mensaje informando de la imposibilidad de realización la acción hasta que la cola tenga menos de 15 elementos.

CASO DE USO #15	**Eliminar un elemento de la cola**	
Objetivo en contexto	Eliminar un nuevo elemento de la cola creada con anterioridad	
Entradas	No hay	
Precondiciones	La cola ha sido creada y existe algún elemento en la cola	
Salidas	Visualización de la cola con un nodo menos al principio de la misma y animación de la eliminación del elemento de la cola. Si no se cumple alguna de las precondiciones se mostrará un mensaje de error.	
Poscondición si éxito	Se elimina el primer elemento de la cola	
Poscondición si fallo	La cola permanece en el mismo estado en que se hallaba cuando se pulsó esta opción, es decir, el elemento no se elimina	
Actores	Usuario de la herramienta o usuario de desarrollo.	
Secuencia normal		
Paso	**Acción**	**Respuesta**
1	Pulsar “Eliminar” desde el menú de herramientas o desde la caja de funciones y pulsando al botón aplicar.	Se comprueba que la cola se ha creado.
1a.	Si la cola se ha creado	Comprobar que la cola no esté vacía, ir a 1c.
1b.	Si la cola no se ha creado	Ir a S1
1c.	Si la cola no está vacía	Se elimina el primer elemento de la cola interna, de manera que reducirá su tamaño en 1. Ir a 2.
1d.	Si la cola está vacía	Ir a S2

2a.	Si la vista seleccionada es vista usuario de la herramienta	Podrá verse de forma animada cómo se elimina el primer elemento de la cola, el cual es el que se encuentra situado más a la derecha de la cola. Esta operación hace que el camión que se sitúa más cerca del surtidor termine de repostar y se vaya. Cuando el primer camión termine de repostar, el segundo camión de la misma pasará a situarse tras el surtidor, en caso de haber más de un elemento en la cola. En caso de que no hubiera más camiones nos encontraríamos ante una gasolinera vacía (sin camiones esperando tras él).
2b1	Si la vista elegida es usuario de desarrollo y se ha elegido la implementación estática.	Se extrae de la cola el primer elemento que figura en la misma, esto es, el que está siendo apuntado por el índice primero. Tras la eliminación, si en la cola hay más elementos, el segundo elemento que figuraba en la misma se convertirá en el primero y pasará a ser apuntado por el índice primero. En caso de que no hubiera más elementos nos encontraremos con un vector vacío en el que los índices primero y último apuntan a la misma posición, a la posición que estaba siendo apuntada anteriormente por libre.
2b2.	Si la vista elegida es usuario de desarrollo y se ha elegido la implementación dinámica.	Al eliminar el primer elemento de la cola, el que está siendo apuntado por el puntero primero, si en la cola hay más elementos, el segundo elemento que figuraba en la misma se convertirá en el primero y pasará a ser apuntado por el puntero primero, es decir, el primer elemento de la cola será el primero→siguiente. En caso de que no hubiera más elementos nos encontraremos ante la cola vacía.
Secuencia alternativa		
Paso	**Acción**	**Respuesta**

S1	Cola no creada	Se muestra un mensaje de error que indique que la cola no se ha creado.
S2	Cola vacía	Se muestra un mensaje informando sobre que no se puede eliminar debido a que la cola no contiene elementos.

CASO DE USO #16	**Consultar el primero**	
Objetivo en contexto	Consultar el primer elemento de la cola.	
Entradas	No hay	
Precondiciones	La cola ha sido creada y exista algún elemento en la cola	
Salidas	La cola permanece en el mismo estado en que se hallaba cuando se pulsó esta opción. Se muestra el primer elemento de la cola.	
Poscondición si éxito	No hay	
Poscondición si fallo	La cola permanece en el mismo estado en que se hallaba cuando se pulsó esta opción. Se muestra un mensaje de error.	
Actores	Usuario de la herramienta o usuario de desarrollo.	
Secuencia normal		
Paso	**Acción**	**Respuesta**
1	Seleccionar la opción "Consultar el primero" del menú herramientas o desde la caja de funciones y pulsando el botón aplicar.	Se comprueba si la cola se ha creado.
1a.	Si la cola se ha creado	Comprobar que la cola tiene algún elemento. Ir a 1c.
1b.	Si la cola no se ha creado	Ir a S1
1c.	Si la cola no está vacía	Se muestra el primer elemento de la cola interna.
1d.	Si la cola está vacía	Ir a S2
Secuencia alternativa		
Paso	**Acción**	**Respuesta**
S1	Cola no creada	Se muestra un mensaje de error que indique que la cola no se ha creado

S2	Cola vacía	Se muestra un mensaje informando sobre que no se puede mostrar el primer elemento debido a que la cola no contiene elementos.

CASO DE USO #17	**Consultar si la cola está vacía**	
Objetivo en contexto	Consultar si la cola está vacía.	
Entradas	No hay	
Precondiciones	La cola ha sido creada	
Salidas	La cola permanece en el mismo estado en que se hallaba cuando se pulsó esta opción. Se muestra si la cola está vacía o no.	
Poscondición si éxito	No hay	
Poscondición si fallo	La cola permanece en el mismo estado en que se hallaba cuando se pulsó esta opción. Se muestra un mensaje de error.	
Actores	Usuario de la herramienta o usuario de desarrollo.	
Secuencia normal		
Paso	**Acción**	**Respuesta**
1	Pulsar “Consultar si la cola está vacía” desde el menú de herramientas o desde la caja de funciones y pulsando el botón aplicar.	Se comprueba que la cola se ha creado.
1a.	Si la cola se ha creado	Se devuelve si la cola está vacía o si no lo está.
1b.	Si la cola no se ha creado	Ir a S1
Secuencia alternativa		
Paso	**Acción**	**Respuesta**
S1	Cola no creada	Se muestra un mensaje de error que indique que la cola no se ha creado

CASO DE USO #18	**Consultar el tamaño de la cola**
Objetivo en contexto	Consultar el tamaño de la cola
Entradas	No hay

Precondiciones	La cola ha sido creada
Salidas	La cola permanece en el mismo estado en que se hallaba cuando se pulsó esta opción. Se devuelve el número de elementos de la cola.
Poscondición si éxito	No hay
Poscondición si fallo	La cola permanece en el mismo estado en que se hallaba cuando se pulsó esta opción. Se muestra un mensaje de error.
Actores	Usuario de la herramienta o usuario de desarrollo.

Secuencia normal		
Paso	**Acción**	**Respuesta**
1	Pulsar "Consultar el tamaño de la cola" desde el menú de herramientas o desde la caja de funciones y pulsando el botón aplicar.	Se comprueba que la cola se ha creado.
1a.	Si la cola se ha creado	Se muestra el número de elementos de la cola.
1b.	Si la cola no se ha creado	Ir a S1

Secuencia alternativa		
Paso	**Acción**	**Respuesta**
S1	Cola no creada	Se muestra un mensaje de error que indique que la cola no se ha creado

5.3. Subsistema de Visualización y Animación del Árbol

Debido a que se usará la misma representación para los árboles AVL y para los árboles binarios de búsqueda no crearemos casos de uso diferenciadores entre ambas estructuras. La única diferencia significativa será la carencia de vista usuario de desarrollo con implementación estática en los árboles AVL.

CASO DE USO #19	**Crear Árbol**
Objetivo en contexto	Crear un árbol
Entradas	Tipo de la estructura que se desea crear.
Precondiciones	El usuario se encuentra en la parte de la aplicación de la estructura de datos árbol.
Salidas	Si no se cumple la precondición se muestra un mensaje de error, si no, se

	dibuja un árbol vacío.
Poscondición si éxito	Se crea una estructura arbórea nueva.
Poscondición si fallo	No se da.
Actores	Usuario de la herramienta o usuario de desarrollo.

Secuencia normal

Paso	Acción	Respuesta
1.	Seleccionar la opción de "Crear" desde el menú de herramientas o desde la caja de funciones y pulsando al botón aplicar.	Mostrar una ventana que pida el tipo de los elementos que contendrá la estructura arbórea.
2.	Seleccionar el tipo de los elementos de la cola.	Se crea una nueva estructura de datos interna árbol binario de búsqueda o árbol AVL dependiendo en la estructura en la que nos hallemos, sobre la cual se realizarán el resto de las operaciones de la estructura arborescente. La implementación interna usada para los árboles será mediante punteros. Si ya existía un árbol creado con anterioridad, se descarta, y se comienza con un nuevo árbol. Se comprueba en qué vista e implementación se encuentra el usuario.
3a.	Si el usuario se encuentra en la vista de usuario de la herramienta.	Se pinta un árbol vacío en el panel y se muestra una etiqueta que informa de que la estructura arbórea ha sido creada. La representación del árbol vacío en esta vista será la de un árbol desnudo (sin hojas)
3b.	Si el usuario se encuentra en el árbol binario de búsqueda en la vista de usuario de desarrollo con la implementación estática.	Se pinta una árbol vacío en el panel cuya representación será una toma de tierra. Se muestra una etiqueta que informa sobre la creación de la cola.

3c.	Si el usuario se encuentra en la vista de usuario con la implementación dinámica.	Se pinta la cola vacía en el panel, cuya representación será cuya representación será una toma de tierra. Se muestra una etiqueta que informa acerca de la creación de la cola.

CASO DE USO #20	**Insertar un elemento en el árbol**	
Objetivo en contexto	Añadir un nuevo elemento a la estructura arbórea creada con anterioridad	
Entradas	Elemento a añadir	
Precondiciones	La estructura arbórea ha sido creada, no tiene completo el último nivel y el elemento es del tipo especificado al crearla.	
Salidas	Se visualiza y anima la inserción del nuevo elemento en el lugar correspondiente según el tipo de árbol elegido y se reordenan los elementos del árbol	
Poscondición si éxito	Se inserta el nuevo elemento en el lugar correspondiente según el tipo de árbol elegido y se reordenan los elementos del árbol	
Poscondición si fallo	El árbol permanece en el mismo estado en que se hallaba cuando se pulsó esta opción, es decir, el elemento no se inserta	
Actores	Usuario de la herramienta o usuario de desarrollo.	
Secuencia normal		
Paso	**Acción**	**Respuesta**
1	Seleccionar la opción ”Insertar” desde el menú de herramientas o desde la caja de funciones y pulsando el botón aplicar. Insertar el elemento en el cuadro de texto activado	Se comprueba que la estructura arbórea se ha creado.
1a.	Si el árbol se ha creado	Comprobar que el último nivel del árbol (nivel 4) no esté lleno. Ir a 1.c
1b.	Si el árbol no se ha creado	Ir a S1
1.c	Si el último nivel del árbol no está lleno	Comprobar que el elemento introducido es del tipo especificado al crear la estructura arbórea, ir a 1.e
1.d	Si el último nivel del árbol está lleno	Ir a S2
1.e	Si el elemento es del tipo especificado	Se añade el elemento a la estructura arbórea interna, ir a 2

1.f	Si el elemento no es del tipo especificado	Ir a S3
2ª	Si el usuario se encuentra en la vista de usuario de la herramienta.	Podrá verse de manera animada cómo se introduce el nuevo elemento en la estructura arbórea en la posición que le corresponde. El elemento insertado se situará en una hoja situada en la copa del árbol ordenadamente, es decir, respetando la propiedad de que para cualquier nodo del árbol su hijo izquierdo es menor que él y su hijo derecho mayor.
2b	Si el usuario se encuentra en el árbol binario de búsqueda en la vista de usuario de desarrollo con la implementación estática.	Se verá cómo se introduce un nuevo elemento en el array que representa al árbol. El elemento se añadirá respetando la propiedad de orden existente entre los nodos de modo que para cualquier nodo del árbol su hijo izquierdo es menor que él y su hijo derecho mayor. El hijo izquierdo de un elemento dado se encontrará en la posición 2^n del vector y el elemento derecho en la 2^{n+1}. Al insertar pueden quedar huecos no rellenos en el vector, esto es debido a que el árbol es no completo.
2c	Si el usuario se encuentra en la vista de usuario con la implementación dinámica.	Podrá verse de manera animada cómo se introduce el nuevo elemento en la estructura arbórea en la posición que le corresponde. El elemento insertado se situará en una hoja situada en tras los últimos niveles ocupados del árbol ordenadamente, es decir, respetando la propiedad de que para cualquier nodo del árbol su hijo izquierdo es menor que él y su hijo derecho mayor.
3	Si la estructura arbórea es un árbol AVL	Se comprueba si el árbol ha quedado desequilibrado tras la inserción, esto es, que para todos los nodos del árbol, las alturas de los subárboles derecho e izquierdo se diferencien a lo sumo en uno. Si el árbol se ha desequilibrado ir a caso de uso equilibrar

Secuencia alternativa		
Paso	**Acción**	**Respuesta**
S1	Árbol no creado	Se muestra un mensaje de error que indique que la estructura arbórea no se ha creado.
S2	Árbol lleno	Se muestra un mensaje que indique que no se puede llevar a cabo la operación debido a que se han superado los límites del árbol.
S3	El tipo del elemento que se desea insertar no se corresponde con el tipo del árbol	Si muestra un mensaje de error informando sobre la incompatibilidad de tipos y no se añade el elemento introducido.

CASO DE USO #21	**Eliminar un elemento del árbol**
Objetivo en contexto	Eliminar un nuevo elemento de la estructura arbórea creada con anterioridad
Entradas	Elemento a eliminar
Precondiciones	El árbol ha sido creado y existe algún elemento en el mismo
Salidas	Se visualizará y animará la eliminación del elemento del árbol y se reordenan los elementos del árbol
Poscondición si éxito	Se elimina el elemento del árbol y se reordenan los elementos del árbol
Poscondición si fallo	La estructura arbórea permanece en el mismo estado en que se hallaba cuando se pulsó esta opción, es decir, el elemento no se elimina
Actores	Usuario de la herramienta o usuario de desarrollo.

Secuencia normal		
Paso	**Acción**	**Respuesta**
1	Pulsar “Eliminar” desde el menú de herramientas o desde la caja de funciones y pulsando al botón aplicar. Introducir el elemento a eliminar en el cuadro de texto.	Se comprueba que la estructura arbórea se ha creado.
1a.	Si el árbol se ha creado	Comprobar que el árbol no esté vacío, ir a 1c.
1b.	Si el árbol no se ha creado	Ir a S1
1c.	Si el árbol no está vacía	Se comprueba que el elemento introducido por el usuario pertenezca al árbol, ir a 1e

1d.	Si la cola está vacía	Ir a S2
1e.	Si el elemento introducido por el usuario pertenece al árbol	Se elimina el elemento de la estructura arbórea interna. Ir a 2.
1f.	Si el elemento introducido por el usuario no pertenece al árbol	Ir a S3
2		Podrá verse de manera animada cómo se elimina el elemento introducido por el usuario. Se buscará el elemento a eliminar y se procederá a su eliminación, reordenando tras la misma los elementos que componen el árbol para que cumplan la propiedad de que para cualquier nodo del árbol su hijo izquierdo es menor que él y su hijo derecho mayor. Si el nodo a eliminar tiene hijo derecho e izquierdo o si sólo tiene hijo derecho, colocaremos en la posición que ocupaba el elemento, el menor elemento del hijo derecho. Y si sólo tiene hijo izquierdo colocaremos éste en la posición del elemento a eliminar. Para visualizar mejor el intercambio se mostrará una flecha indicando el elemento que va a sustituir al anterior. Tras mostrar la sustitución que se va a llevar a cabo se mostrará una X tachando al elemento que se va a eliminar y se extraerá animadamente el mismo. Si la estructura arbórea sólo constaba del elemento que se va a suprimir, el elemento se borrará y se mostrará una toma a tierra.
3	Si la estructura arbórea es un árbol AVL	Se comprueba si el árbol ha quedado desequilibrado tras la eliminación, esto es, que para todos los nodos del árbol, las alturas de los subárboles derecho e izquierdo se diferencien a lo sumo en uno. Si el árbol se ha desequilibrado ir a caso de uso equilibrar
Secuencia alternativa		
Paso	**Acción**	**Respuesta**

S1	Árbol no creado	Se muestra un mensaje de error que indique que el árbol no se ha creado.
S2	Árbol vacío	Se muestra un mensaje informando sobre que no se puede eliminar debido a que el elemento no existe.
S3	El elemento a eliminar no existe	Se mostrará un mensaje informando sobre que no se puede eliminar debido a que el elemento no existe.

CASO DE USO #22	**Recorrer en preorden**	
Objetivo en contexto	Se realizará un recorrido por los elementos del árbol en preorden	
Entradas	No hay	
Precondiciones	La estructura arbórea ha sido creada	
Salidas	La estructura arbórea permanece en el mismo estado en que se hallaba cuando se pulsó esta opción. Se muestra el recorrido.	
Poscondición si éxito	No hay	
Poscondición si fallo	La estructura arbórea permanece en el mismo estado en que se hallaba cuando se pulsó esta opción. Se mostrará un mensaje de error.	
Actores	Usuario de la herramienta o usuario de desarrollo.	
Secuencia normal		
Paso	**Acción**	**Respuesta**
1	Pulsar “recorrer en preorden” desde el menú de herramientas o desde la caja de funciones y pulsando al botón aplicar.	Se comprueba que el árbol se ha creado.
1ª.	Si el árbol se ha creado	Se comprueba que el árbol no sea vacío
1b.	Si el árbol no se ha creado	Ir a S1

1c	Si el árbol no es vacío	Se muestra animadamente el recorrido en preorden del árbol. El recorrido será el recorrido es el elemento raíz, el recorrido en preorden del hijo izquierdo y el recorrido en preorden del hijo derecho. En la animación se iluminarán uno a uno la secuencia de elementos que componen el recorrido.
1d.	Si el árbol es vacío	Ir a S2
Secuencia alternativa		
Paso	**Acción**	**Respuesta**
S1	Árbol no creado	Se muestra un mensaje de error que indique que el árbol no se ha creado.
S2	Árbol vacío	Se muestra un mensaje informando de que no se puede hacer un recorrido de un árbol vacío

CASO DE USO #23	**Recorrer en postorden**	
Objetivo en contexto	Se realizará un recorrido por los elementos del árbol en postorden	
Entradas	No hay	
Precondiciones	La estructura arbórea ha sido creada	
Salidas	La estructura arbórea permanece en el mismo estado en que se hallaba cuando se pulsó esta opción. Se muestra el recorrido.	
Poscondición si éxito	No hay	
Poscondición si fallo	La estructura arbórea permanece en el mismo estado en que se hallaba cuando se pulsó esta opción. Se mostrará un mensaje de error.	
Actores	Usuario de la herramienta o usuario de desarrollo.	
Secuencia normal		
Paso	**Acción**	**Respuesta**
1	Pulsar “recorrer en postorden” desde el menú de herramientas o desde la caja de funciones y pulsando al botón aplicar.	Se comprueba que el árbol se ha creado.
1a.	Si el árbol se ha creado	Se comprueba que el árbol no sea vacío.
1b.	Si el árbol no se ha creado	Ir a S1

1c	Si el árbol no es vacío	Se muestra animadamente el recorrido en postorden del árbol. El recorrido será el recorrido en postorden del hijo izquierdo, el recorrido en postorden del hijo derecho y elemento raíz. En la animación se iluminarán uno a uno la secuencia de elementos que componen el recorrido.
1d.	Si el árbol es vacío	Ir a S2
Secuencia alternativa		
Paso	**Acción**	**Respuesta**
S1	Árbol no creado	Se muestra un mensaje de error que indique que el árbol no se ha creado.
S2	Árbol vacío	Se muestra un mensaje informando de que no se puede hacer un recorrido de un árbol vacío

CASO DE USO #24	**Recorrer en inorden**	
Objetivo en contexto	Consultar el primer elemento de la cola.	
Entradas	No hay	
Precondiciones	La estructura arbórea ha sido creada	
Salidas	La estructura arbórea permanece en el mismo estado en que se hallaba cuando se pulsó esta opción. Se muestra el recorrido. No hay	
Poscondición si éxito	No hay	
Poscondición si fallo	La estructura arbórea permanece en el mismo estado en que se hallaba cuando se pulsó esta opción. Se mostrará un mensaje de error.	
Actores	Usuario de la herramienta o usuario de desarrollo.	
Secuencia normal		
Paso	**Acción**	**Respuesta**
1	Pulsar “recorrer en inorden” desde el menú de herramientas o desde la caja de funciones y pulsando al botón aplicar.	Se comprueba que el árbol se ha creado.
1a.	Si el árbol se ha creado	Se comprueba que el árbol no sea vacío.
1b.	Si el árbol no se ha creado	Ir a S1

1c	Si el árbol no es vacío	Se muestra animadamente el recorrido en inorden del árbol. El recorrido será el inorden del hijo izquierdo, seguido de la raíz y del recorrido en inorden del hijo derecho. En la animación se iluminarán uno a uno la secuencia de elementos que componen el recorrido.
1d.	Si el árbol es vacío	Ir a S2
Secuencia alternativa		
Paso	**Acción**	**Respuesta**
S1	Árbol no creado	Se muestra un mensaje de error que indique que el árbol no se ha creado.
S2	Árbol vacío	Se muestra un mensaje informando de que no se puede hacer un recorrido de un árbol vacío

CASO DE USO #25	**Consultar si el árbol está vacío**	
Objetivo en contexto	Consultar si el árbol está vacío.	
Entradas	No hay	
Precondiciones	La estructura arbórea ha sido creada	
Salidas	La estructura arbórea permanece en el mismo estado en que se hallaba cuando se pulsó esta opción. Se muestra si el árbol está vacío o no.	
Poscondición si éxito	No hay	
Poscondición si fallo	La estructura arbórea permanece en el mismo estado en que se hallaba cuando se pulsó esta opción. Se mostrará un mensaje de error.	
Actores	Usuario de la herramienta o usuario de desarrollo.	
Secuencia normal		
Paso	**Acción**	**Respuesta**
1	Pulsar “Consultar si el árbol está vacía” desde el menú de herramientas o desde la caja de funciones y pulsando al botón aplicar.	Comprobar que el árbol se ha creado.
1a.	Si el árbol se ha creado	Se muestra si la estructura arbórea está vacía o si no lo está
1b.	Si el árbol no se ha creado	Ir a S1

Secuencia alternativa		
Paso	**Acción**	**Respuesta**
S1	Árbol no creado	Se muestra un mensaje de error que indique que el árbol no se ha creado.

CASO DE USO #26	**Consultar los elementos de un nivel del árbol**
Objetivo en contexto	Consultar los elementos de un nivel del árbol
Entradas	No hay
Precondiciones	La estructura arbórea ha sido creada
Salidas	La estructura arbórea permanece en el mismo estado en que se hallaba cuando se pulsó esta opción. Se devuelven los elementos de un nivel del árbol.
Poscondición si éxito	No hay
Poscondición si fallo	La estructura arbórea permanece en el mismo estado en que se hallaba cuando se pulsó esta opción. Se mostrará un mensaje de error.
Actores	Usuario de la herramienta o usuario de desarrollo.

Secuencia normal		
Paso	**Acción**	**Respuesta**
1	Pulsar “Nivel” desde el menú de herramientas o desde la caja de funciones y pulsando al botón aplicar.	Se comprueba que el árbol se ha creado
1a.	Si el árbol se ha creado	Se comprueba que el nivel introducido pertenezca al árbol
1b.	Si el árbol no se ha creado	Ir a S1
1c	Si el nivel introducido pertenece al árbol	Se iluminan los elementos pertenecientes a ese nivel del árbol
1d	Si el nivel introducido no pertenece al árbol	Ir a S2

Secuencia alternativa		
Paso	**Acción**	**Respuesta**
S1	Árbol no creado	Se muestra un mensaje de error que indique que el árbol no se ha creado.

S2	Nivel no existente	Se muestra un mensaje de error informando que el nivel introducido no existe

CASO DE USO #27	**Consultar la altura del árbol**	
Objetivo en contexto	Consultar la altura del árbol	
Entradas	No hay	
Precondiciones	La estructura arbórea ha sido creada	
Salidas	La estructura arbórea permanece en el mismo estado en que se hallaba cuando se pulsó esta opción. Se devuelve la altura del árbol.	
Poscondición si éxito	No hay	
Poscondición si fallo	La estructura arbórea permanece en el mismo estado en que se hallaba cuando se pulsó esta opción. Se mostrará un mensaje de error.	
Actores	Usuario de la herramienta o usuario de desarrollo.	
Secuencia normal		
Paso	**Acción**	**Respuesta**
1	Pulsar “Consultar la altura del árbol” desde el menú de herramientas o desde la caja de funciones y pulsando al botón aplicar.	Se comprueba que el árbol ha sido creado.
1a.	Si el árbol se ha creado	Se muestra la altura del árbol.
1b.	Si el árbol no se ha creado	Ir a S1
Secuencia alternativa		
Paso	**Acción**	**Respuesta**
S1	Árbol no creado	Se muestra un mensaje de error que indique que el árbol no se ha creado.

CASO DE USO #28	**Consultar el hijo izquierdo**
Objetivo en contexto	Consultar el hijo izquierdo de un elemento del árbol
Entradas	Posición del elemento
Precondiciones	La estructura arbórea ha sido creada
Salidas	La estructura arbórea permanece en el mismo estado en que se hallaba cuando se pulsó esta opción. Se muestra el hijo izquierdo del árbol.

Poscondición si éxito	No hay
Poscondición si fallo	La estructura arbórea permanece en el mismo estado en que se hallaba cuando se pulsó esta opción. Se mostrará un mensaje de error.
Actores	Usuario de la herramienta o usuario de desarrollo.

Secuencia normal		
Paso	**Acción**	**Respuesta**
1	Pulsar "Consultar el hijo izquierdo" desde el menú de herramientas o desde la caja de funciones y pulsando al botón aplicar.	Se comprueba que el árbol ha sido creado
1a.	Si el árbol se ha creado	Se iluminará el hijo izquierdo de la raíz. Si el nodo no tuviera hijo izquierdo se mostrará un mensaje indicando que no existe hijo izquierdo para tal nodo.
1b.	Si el árbol no se ha creado	Ir a S1

Secuencia alternativa

Paso	**Acción**	**Respuesta**
S1	Árbol no creado	Se muestra un mensaje de error que indique que el árbol no se ha creado.

CASO DE USO #29	**Consultar el hijo derecho**
Objetivo en contexto	Consultar el hijo derecho de un elemento del árbol
Entradas	Posición del elemento
Precondiciones	La estructura arbórea ha sido creada
Salidas	La estructura arbórea permanece en el mismo estado en que se hallaba cuando se pulsó esta opción. Se muestra el hijo derecho del árbol.
Poscondición si éxito	No hay
Poscondición si fallo	La estructura arbórea permanece en el mismo estado en que se hallaba cuando se pulsó esta opción. Se mostrará un mensaje de error.
Actores	Usuario de la herramienta o usuario de desarrollo.

Secuencia normal		
Paso	**Acción**	**Respuesta**

1	Pulsar "Consultar el hijo derecho" desde el menú de herramientas o desde la caja de funciones y pulsando al botón aplicar.	Se comprueba que el árbol ha sido creado
1a.	Si el árbol se ha creado	Se iluminará el hijo derecho dela raíz. Si el nodo no tuviera hijo derecho se mostrará un mensaje indicando que no existe hijo derecho para tal nodo.
1b.	Si el árbol no se ha creado	Ir a S1
Secuencia alternativa		
Paso	**Acción**	**Respuesta**
S1	Árbol no creado	Se muestra un mensaje de error que indique que el árbol no se ha creado.

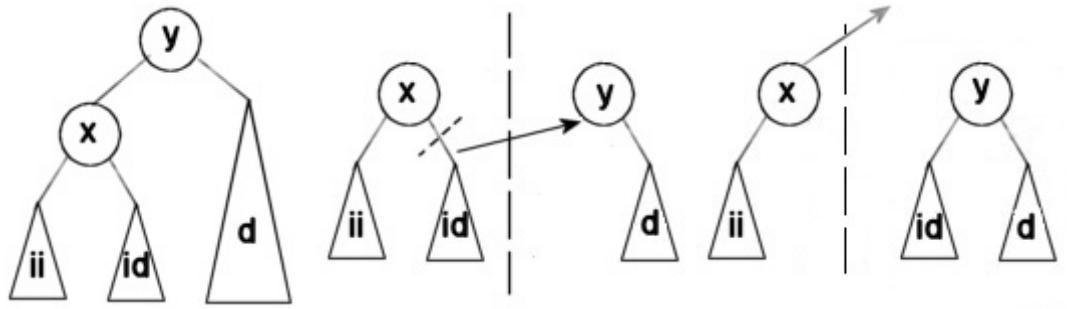
CASO DE USO #30	**Consultar la raíz**	
Objetivo en contexto	Consultar la raíz de un elemento del árbol	
Entradas	Posición del elemento	
Precondiciones	La estructura arbórea ha sido creada	
Salidas	La estructura arbórea permanece en el mismo estado en que se hallaba cuando se pulsó esta opción. Se muestra la raíz.	
Poscondición si éxito	No hay	
Poscondición si fallo	La estructura arbórea permanece en el mismo estado en que se hallaba cuando se pulsó esta opción. Se mostrará un mensaje de error.	
Actores	Usuario de la herramienta o usuario de desarrollo.	
Secuencia normal		
Paso	**Acción**	**Respuesta**
1	Pulsar "Consultar la raíz".	Se comprueba que el árbol ha sido creado
1a.	Si el árbol se ha creado	Se devolverá el valor del elemento situado en el primer nivel del árbol.
1b.	Si el árbol no se ha creado	Ir a S1
Secuencia alternativa		
Paso	**Acción**	**Respuesta**
S1	Árbol no creado	Se muestra un mensaje de error que indique que el árbol no se ha creado.

CASO DE USO #31	**Equilibrar**
Objetivo en contexto	Equilibrar el árbol de modo que para todos los nodos del árbol, las alturas de los subárboles derecho e izquierdo se diferencien a lo sumo en uno.
Entradas	No hay.
Precondiciones	El árbol se encuentra desequilibrado.
Salidas	Se devuelve el árbol equilibrado.
Poscondición si éxito	No hay
Poscondición si fallo	No hay
Actores	No hay

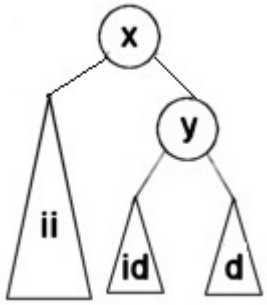
Secuencia normal		
Paso	**Acción**	**Respuesta**
1a.	Si la altura del hijo izquierdo es igual a la altura del hijo derecho más 2.	Nos encontramos ante un desequilibrio izquierdo, ir a 1c.
1b.	Si la altura del hijo derecho es igual a la altura del hijo izquierdo más 2.	Nos encontramos ante un desequilibrio izquierdo, ir a
1c.	Examinar si la rotación izquierda es simple o doble.	Si la altura del hijo izquierdo del hijo izquierdo es mayor o igual que la altura del hijo derecho del hijo izquierdo, la rotación es simple, ir a 2 a, sino ir a 2b
1c.	Examinar si la rotación derecha es simple o doble.	Si la altura del hijo derecho del hijo derecho es mayor o igual que la altura del hijo izquierdo del hijo derecho, la rotación es simple, ir a 2c , sino ir a 2b
2a.	Rotación izquierda simple	

Se dibujará la animación del equilibrio la siguiente manera:

1 2 3

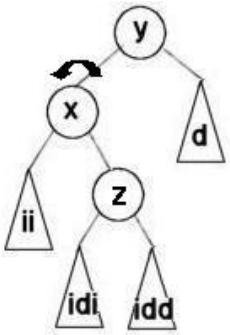


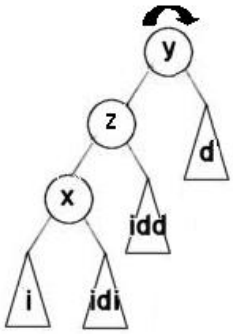
4

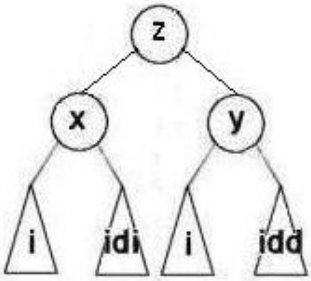
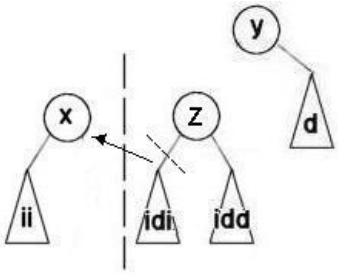


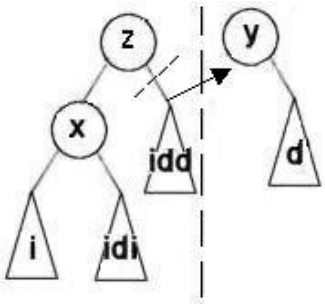
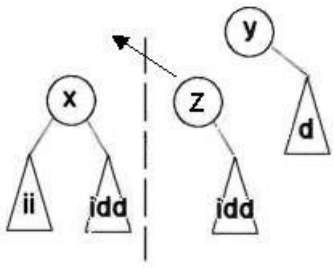
2b.	Rotación izquierda doble

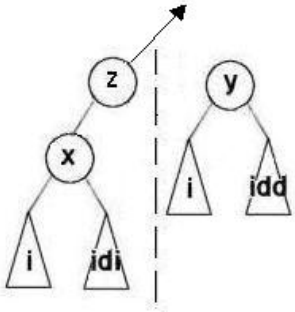
Se dibujará la animación del equilibrio la siguiente manera:

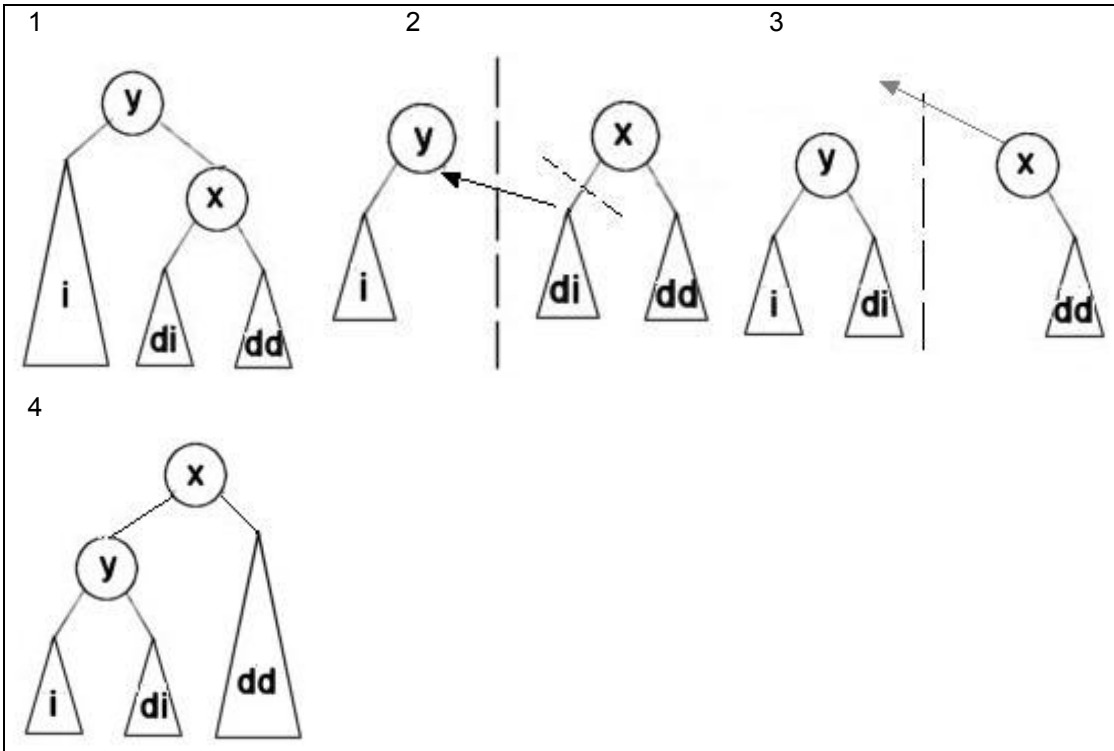
<table>
<tr>
<td>1
 

4
 

7
 
</td>
<td>2
 

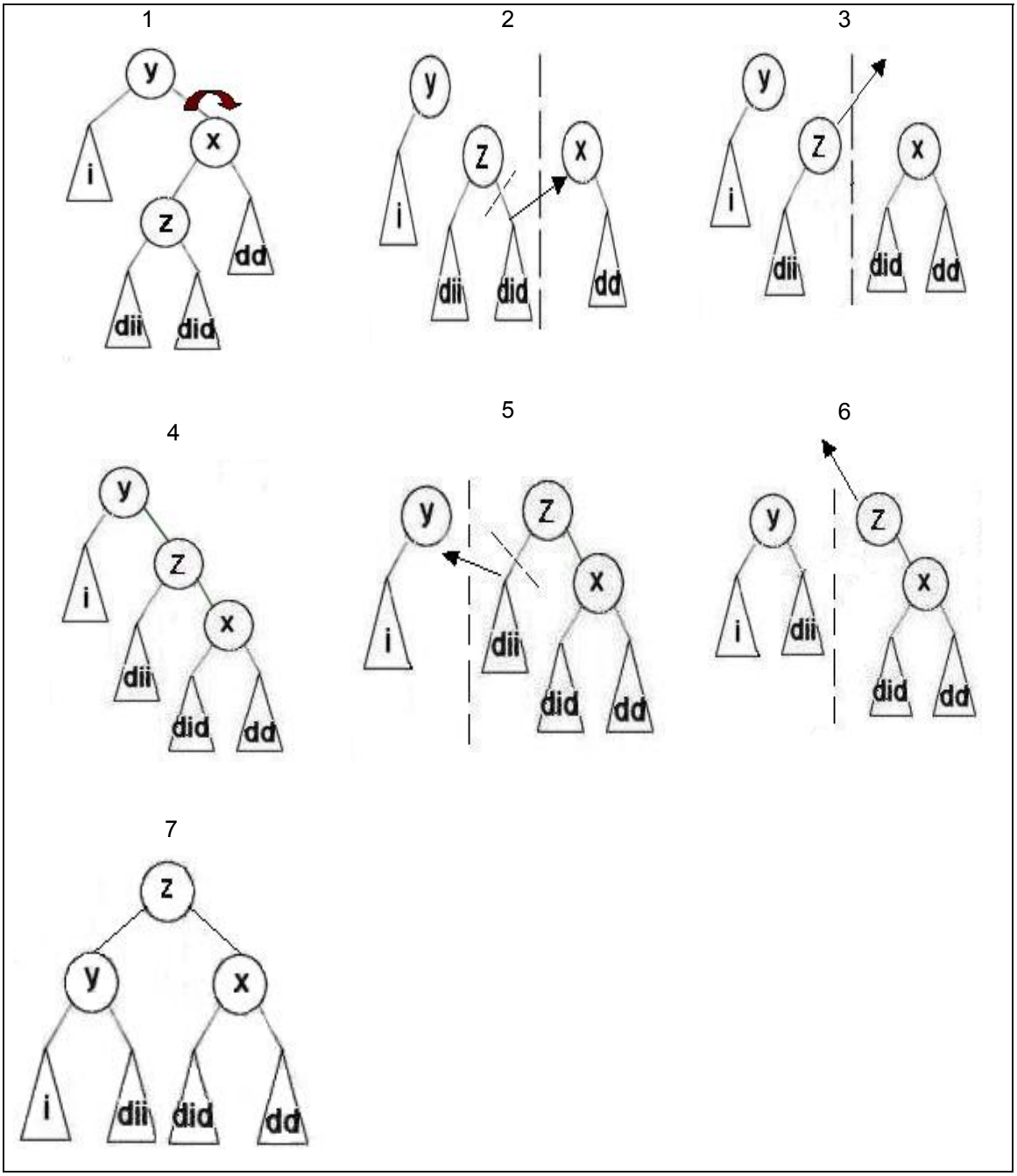
5
 
</td>
<td>3
 

6
 
</td>
</tr>
<tr><td>2c.</td><td colspan="2">Rotación derecha simple</td></tr>
</table>



2d.	Rotación derecha doble

Se dibujará la animación del equilibrio la siguiente manera:



5.4. Visualización y Animación de la Cola de Prioridad

CASO DE USO #32	**Crear Cola de Prioridad**
Objetivo en contexto	Crear una estructura de datos cola de prioridad
Entradas	Tipo de la estructura que se desea crear.
Precondiciones	El usuario se encuentra en la parte de la aplicación de la estructura de

	datos cola de prioridad.
Salidas	Si no se cumple la precondición se muestra un mensaje de error, si no, se dibuja una toma de tierra que representa la cola de prioridad vacía.
Poscondición si éxito	Se crea una cola de prioridad nueva.
Poscondición si fallo	Se muestra un mensaje de error, que informa de la razón del fallo.
Actores	Usuario de la herramienta o usuario de desarrollo.

Secuencia normal

Paso	Acción	Respuesta
1.	Seleccionar la opción de "Crear" desde el menú de herramientas o desde la caja de funciones y pulsando al botón aplicar. Tras la selección se solicitará el tipo de los elementos que contendrá la cola de prioridad.	Se crea una nueva estructura de datos cola de prioridad interna, sobre la cual se realizarán el resto de las operaciones de la estructura. En cualquiera de las dos vistas se muestra una cola de prioridad vacía, cuya representación será una toma a tierra.

CASO DE USO #33	**Insertar elemento a cola de prioridad**
Objetivo en contexto	Añadir un nuevo elemento a la cola de prioridad creada con anterioridad
Entradas	Elemento a añadir.
Precondiciones	La cola de prioridad ha sido creada y el elemento es del tipo especificado al crearla
Salidas	Visualización de la cola de prioridad con un nuevo nodo situado en la posición adecuada de la cola de prioridad y animación de la inserción del elemento en la cola de prioridad. Si no se cumple alguna de las precondiciones se mostrará un mensaje de error.
Poscondición si éxito	Se inserta el nuevo elemento en la posición adecuada de la cola de prioridad de forma que se mantenga la propiedad de orden de la cola de prioridad, esto es, el valor de cada nodo interno es menor o igual que los valores de sus hijos.
Poscondición si fallo	La cola de prioridad permanece en el mismo estado en que se hallaba cuando se pulsó esta opción, es decir, el elemento no se inserta
Actores	Usuario de la herramienta o usuario de desarrollo.

Secuencia normal

Paso	Acción	Respuesta

1	Seleccionar la opción de “Insertar” desde el menú de herramientas o desde la caja de funciones y pulsando al botón aplicar. Insertar el elemento en el cuadro de texto activado.	Se comprueba que la cola de prioridad se ha creado.
1.a	Si la cola de prioridad se ha creado	Si el usuario se encuentra en la vista estática comprobar que la cola no esté llena, ir a 1.c, sino ir a 1e.
1.b	Si la cola de prioridad no se ha creado	Ir a S1
1.c	Si la cola de prioridad no está llena	Comprobar que el elemento introducido es del tipo especificado al crear la cola, ir a 1.e
1.d	Si la cola de prioridad está llena	Ir a S2
1.e	Si el elemento es del tipo especificado	Se añade el elemento a la cola de prioridad interna de manera que aumentará su tamaño en 1, ir a 2
1.f	Si el elemento no es del tipo especificado	Ir a S3
2a.	Si la vista seleccionada es la de usuario de la herramienta	Al insertar un elemento podrá verse cómo el elemento introducido por el usuario se sitúa al final de los nodos previamente insertados. Si el elemento insertado posee más prioridad que el resto se actualizará la flecha que señala el mínimo. Puede observarse que el valor del elemento aparece centrado en el nodo y que el tamaño del nodo se adapta al tamaño del texto introducido.

2b.	Si la vista elegida es la de usuario de desarrollo y se encuentra en la implementación estática del montículo	Se podrá ver cómo se sitúa en la posición adecuada del montículo, de manera que no se pierda la relación de orden. Si al realizarse la inserción deja de ser una cola de prioridad se deberá flotar el elemento hacia la primera posición de la cola de prioridad, esto es, intercambiar el elemento con su padre, hasta que se cumpla la propiedad de la cola de prioridad. Para visualizar la animación de la inserción primero haremos sitio al elemento mostrando los intercambios necesarios entre hijo-padre mediante flechas y posteriormente introduciremos el elemento animadamente por la izquierda en la posición que le corresponda.
2c.	Si la vista elegida es la de usuario de desarrollo y se encuentra en la visualización arbórea del montículo	Se visualizará cómo se sitúa en la posición adecuada el elemento introducido en la cola de prioridad, de manera que no se pierda la relación de orden. Si al realizarse la inserción deja de ser una cola de prioridad se deberá flotar el elemento hacia la raíz, esto es, intercambiar el elemento con su padre, hasta que se cumpla la propiedad de la cola de prioridad. Para representar la inserción primero haremos sitio al elemento mostrando los intercambios necesarios mediante flechas y posteriormente introduciremos el elemento animadamente.
Secuencia alternativa		
Paso	**Acción**	**Respuesta**
S1	La cola de prioridad no se ha creado	Se muestra un mensaje de error que indique que la cola de prioridad no se ha creado.
S2	La cola de prioridad está llena	Se muestra un mensaje de error informando sobre la incompatibilidad de tipos y no se añade el elemento introducido.

S3	El elemento que se desea insertar en la cola de prioridad no es del tipo especificado.	Se muestra un mensaje que indique que no se puede llevar a cabo la operación debido a que se han superado los límites de la cola de prioridad.

CASO DE USO #34	**Eliminar elemento mínimo de la cola de prioridad**	
Objetivo en contexto	Eliminar el elemento con mayor prioridad, esto es, el de menor valor de la cola de prioridad creada con anterioridad	
Entradas	No hay	
Precondiciones	La cola de prioridad ha sido creada y existe algún elemento en la ella	
Salidas	Se visualiza y anima la extracción del elemento mínimo de la cola de prioridad.	
Poscondición si éxito	Se elimina el elemento de mayor prioridad de la cola de prioridad.	
Poscondición si fallo	La cola de prioridad permanece en el mismo estado en que se hallaba cuando se pulsó esta opción, es decir, el elemento no se elimina	
Actores	Usuario de la herramienta o usuario de desarrollo.	
Secuencia normal		
Paso	**Acción**	**Respuesta**
1	Pulsar “Eliminar” desde el menú de herramientas o desde la caja de funciones y pulsando al botón aplicar.	Se comprueba que la cola de prioridad se ha creado.
1a.	Si la cola de prioridad se ha creado	Comprobar que la cola de prioridad no esté vacía, ir a 1c.
1b.	Si la cola de prioridad no se ha creado	Ir a S1
1c.	Si la cola de prioridad no está vacía	Se elimina el elemento mínimo de la cola de prioridad interna, de manera que reducirá su tamaño en 1. Ir a 2.
1d.	Si la cola de prioridad está vacía	Ir a S2
2a.	Si la vista seleccionada es vista usuario de la herramienta	Podrá verse de forma animada cómo se elimina mínimo de la cola de prioridad, actualizándose el nuevo mínimo. En caso de que no hubiera más elementos se pintaría la cola de prioridad vacía (una toma a tierra).

2b.	Si la vista elegida es la de usuario de desarrollo y se encuentra en la implementación estática del montículo	Se eliminará el elemento de mayor prioridad desplazándose para ello los elementos que sea necesario. Y posteriormente actualizando el mínimo, mostrando los intercambios entre elementos realizados al hundir a través de flechas.
2c.	Si la vista elegida es la de usuario de desarrollo y se encuentra en la visualización arbórea del montículo	En la eliminación del elemento mínimo se observará de manera animada cómo el elemento de mayor prioridad (aquel que se sitúa en la raíz del montículo) desaparece disminuyendo el tamaño de la estructura de datos. Para llevar a cabo la eliminación del elemento mínimo, se colocará el último elemento en la raíz y se aplicará la función de hundir hasta que se cumpla la propiedad de la cola de prioridad. Esta operación consistirá en intercambiar el elemento con su hijo más pequeño. Los intercambios entre elementos se mostrarán a través de flechas.
Secuencia alternativa		
Paso	**Acción**	**Respuesta**
S1	La cola de prioridad no se ha creado	Se muestra un mensaje de error que indique que la cola de prioridad no se ha creado.
S2	La cola de prioridad está vacía	Se muestra un mensaje informando que no se puede eliminar debido a que la cola de prioridad no contiene elementos.

CASO DE USO #35	**Consultar elemento mínimo de la cola de prioridad**
Objetivo en contexto	Consultar el elemento mínimo de la cola de prioridad.
Entradas	No hay
Precondiciones	La cola de prioridad ha sido creada y exista algún elemento en la ella
Salidas	Se visualiza en una etiqueta de la interfaz el valor del elemento mínimo.
Poscondición si éxito	La cola de prioridad permanece en el mismo estado en que se hallaba cuando se pulsó esta opción.
Poscondición si	La cola de prioridad permanece en el mismo estado en que se hallaba

fallo	cuando se pulsó esta opción. Se muestra un mensaje de error.	
Actores	Usuario de la herramienta o usuario de desarrollo.	
Secuencia normal		
Paso	**Acción**	**Respuesta**
1	Seleccionar la opción "Consultar mínimo" del menú herramientas o desde la caja de funciones y pulsando el botón aplicar.	Se comprueba que la cola de prioridad ha sido creada.
1a.	Si la cola de prioridad se ha creado	Comprobar que la cola de prioridad tiene algún elemento. Ir a 1c.
1b.	Si la cola de prioridad no se ha creado	Ir a S1
1c.	Si la cola de prioridad no está vacía	Se muestra el elemento mínimo de la cola de prioridad.
1d.	Si la cola de prioridad está vacía	Ir a S2
Secuencia alternativa		
Paso	**Acción**	**Respuesta**
S1	La cola de prioridad no se ha creado	Se muestra un mensaje de error que indique que la cola de prioridad no se ha creado
S2	La cola de prioridad está vacía	Se muestra un mensaje informando que no se puede mostrar el primer elemento debido a que la cola de prioridad no contiene elementos.

CASO DE USO #36	**Consultar si la cola de prioridad está vacía**	
Objetivo en contexto	Consultar si la cola de prioridad está vacía.	
Entradas	No hay	
Precondiciones	La cola de prioridad ha sido creada	
Salidas	Se visualiza un mensaje que informe si la cola de prioridad está vacía o no	
Poscondición si éxito	La cola de prioridad permanece en el mismo estado en que se hallaba cuando se pulsó esta opción.	
Poscondición si fallo	La cola de prioridad permanece en el mismo estado en que se hallaba cuando se pulsó esta opción. Se muestra un mensaje de error.	
Actores	Usuario de la herramienta o usuario de desarrollo.	
Secuencia normal		
Paso	**Acción**	**Respuesta**

1	Seleccionar la opción de "¿Está Vacía?" desde el menú de herramientas o desde la caja de funciones y pulsando el botón aplicar.	Se comprueba que la cola de prioridad se ha creado.
1a.	Si la cola de prioridad se ha creado	Se indica en la etiqueta de la interfaz si la cola de prioridad está vacía o si no lo está.
1b.	Si la cola de prioridad no se ha creado	Ir a S1
Secuencia alternativa		
Paso	**Acción**	**Respuesta**
S1	La cola de prioridad no se ha creado	Se muestra un mensaje de error que indique que la cola de prioridad no se ha creado

CASO DE USO #37	**Consultar el tamaño de la cola de prioridad**	
Objetivo en contexto	Consultar el tamaño de la cola de prioridad	
Entradas	No hay	
Precondiciones	La cola de prioridad ha sido creada	
Salidas	Se visualiza en la etiqueta de la interfaz el tamaño de la cola de prioridad	
Poscondición si éxito	La cola de prioridad permanece en el mismo estado en que se hallaba cuando se pulsó esta opción. Se devuelve el número de elementos de la cola de prioridad.	
Poscondición si fallo	La cola de prioridad permanece en el mismo estado en que se hallaba cuando se pulsó esta opción. Se muestra un mensaje de error.	
Actores	Usuario de la herramienta o usuario de desarrollo.	
Secuencia normal		
Paso	**Acción**	**Respuesta**
1	Seleccionar la opción "Tamaño" desde el menú de herramientas o desde la caja de funciones y pulsando el botón aplicar.	Se comprueba que la cola de prioridad se ha creado.
1a.	Si la cola de prioridad se ha creado	Se muestra el número de elementos de la cola de prioridad.
1b.	Si la cola de prioridad no se ha creado	Ir a S1
Secuencia alternativa		
Paso	**Acción**	**Respuesta**

S1	La cola de prioridad no se ha creado	Se muestra un mensaje de error que indique que la cola de prioridad no se ha creado

6. Documentación de las Estructuras de Datos

CASO DE USO #38	**Mostrar especificación de la estructura de datos**
Objetivo en contexto	Consultar la especificación de la estructura de datos, la cual indica las operaciones que pueden realizarse sobre la estructura, de manera independiente de la implementación.
Entradas	Ninguna.
Precondiciones	El usuario se encuentra en la ventana principal de la estructura de datos de la que se quiere obtener la especificación.
Salidas	Especificación de la estructura de datos.
Poscondición si éxito	No hay.
Poscondición si fallo	Ninguna.
Actores	Usuario de la herramienta o usuario de desarrollo.

Secuencia normal

Paso	**Acción**	**Respuesta**
1.	Seleccionar desde el menú de herramientas la opción “Especificación”.	Mostrar en una ventana aparte una página de documentación HTML que muestre la especificación de la pila.

CASO DE USO #39	**Mostrar código fuente de la estructura de datos**
Objetivo en contexto	Se mostrará el código con el que se ha implementado la estructura, dependiendo de la implementación escogida en la vista usuario de desarrollo. En el caso de que se haya escogido la vista usuario de la herramienta, se mostrará el código de todas las implementaciones posibles.
Entradas	Ninguna.
Precondiciones	El usuario se encuentra en la ventana principal de la estructura de datos de la que se quiere obtener el código.

Salidas	Código fuente en Java de la estructura de datos, en función de la implementación escogida.
Poscondición si éxito	No hay.
Poscondición si fallo	Ninguna.
Actores	Usuario de la herramienta o usuario de desarrollo.

Secuencia normal

Paso	**Acción**	**Respuesta**
1.	Seleccionar desde el menú de herramientas la opción “Código”.	Comprobar la vista en la que se encuentra el usuario.
2a.	Si el usuario se encuentra en la vista de usuario de desarrollo y ha escogido la implementación estática.	Mostrar en una ventana aparte una página de documentación HTML que muestre el código con una implementación estática de la estructura de datos.
2b.	Si el usuario se encuentra en la vista de usuario de desarrollo y ha escogido la implementación dinámica.	Mostrar en una ventana aparte una página de documentación HTML que muestre el código con una implementación dinámica de la estructura de datos.
2c.	Si el usuario se encuentra en la vista de usuario de la herramienta.	Mostrar en una ventana aparte una página de documentación HTML que muestre el código con todas las implementaciones posibles de la estructura de datos.

CASO DE USO #40	Mostrar coste de la estructura de datos
Objetivo en contexto	Visualizar una tabla que permita ver el coste de las operaciones de la estructura de datos en las diferentes implementaciones.
Entradas	Ninguna.
Precondiciones	El usuario se encuentra en la ventana principal de la estructura de datos de la que se quiere obtener el coste.
Salidas	Visualización de una tabla que permita comparar el coste asintótico temporal y espacial de cada una de las operaciones de la estructura de datos escogida en las diferentes implementaciones.

Poscondición si éxito	No hay
Poscondición si fallo	Ninguna.
Actores	Usuario de la herramienta o usuario de desarrollo.

Secuencia normal

Paso	Acción	Respuesta
1.	Seleccionar desde el menú de herramientas la opción “Coste”.	Mostrar en una ventana aparte una página de documentación HTML que muestre una tabla que permita comparar el coste asintótico temporal y espacial de cada una de las operaciones de la estructura de datos escogida en las diferentes implementaciones.

CASO DE USO #41	Mostrar ayuda adicional de la estructura de datos
Objetivo en contexto	Se dispondrá de una página de ayuda que contenga información general sobre la estructura de datos (definición de la estructura de datos, características, operaciones permitidas, etc).
Entradas	Ninguna.
Precondiciones	El usuario se encuentra en la ventana principal de la estructura de datos de la que se quiere obtener la ayuda adicional.
Salidas	Página de ayuda con información general sobre la estructura de datos.
Poscondición si éxito	No hay
Poscondición si fallo	Ninguna.
Actores	Usuario de la herramienta o usuario de desarrollo.

Secuencia normal

Paso	Acción	Respuesta

1.	Seleccionar desde el menú de herramientas la opción "Ayuda Adicional".	Mostrar en una ventana aparte una página de documentación HTML que muestre la información general sobre la estructura de datos (definición de la estructura de datos, características, operaciones permitidas, etc).

7. Visualización y Animación de Esquemas Algorítmico

CASO DE USO #42	**Introducir datos**	
Objetivo en contexto	Introducción de los datos, por parte del usuario, con los que quiere que se ejecute y visualice el algoritmo escogido.	
Entradas	Los datos del problema.	
Precondiciones	El algoritmo a ejecutar se seleccionó con anterioridad, y es la primera que se selecciona o se pulsa nuevos datos.	
Salidas	Si se cumple las precondiciones se piden los datos necesarios para el problema.	
Poscondición si éxito	Se introducen en el ejemplo los datos proporcionados por el usuario.	
Poscondición si fallo	Se muestra un mensaje de error, que informa de la razón del fallo.	
Actores	Usuario de la herramienta o usuario de desarrollo.	
Secuencia normal		
Paso	**Acción**	**Respuesta**
1.	Seleccionar el algoritmo con el que se ejecutará el esquema algorítmico o pulsar el botón "Nuevos datos".	Se muestra los datos que se requerirán en el algoritmo. Se visualiza el panel que solicita los datos de entrada del problema.

2.	Introducción de los datos: En el caso del algoritmo de la mochila el usuario deberá introducir el número de objetos, la capacidad máxima de peso de la mochila y el valor y el peso de cada objeto. En el caso del algoritmo de búsqueda binaria debe introducirse la longitud del vector, el vector con sus elementos y el elemento a buscar. En el caso del algoritmo de Quicksort deberá introducirse el número de elementos a ordenar y el valor de cada uno de ellos. En el caso del algoritmo de Dijkstra deberá introducirse el número de nodos, la distancia existente entre ellos.	Se comprueba la corrección de los datos introducidos por el usuario.
3a.	Si los datos no son correctos.	Ir a S1.
3b.	Si los datos son correctos.	Se introducen los datos internamente en el ejemplo escogido.
Secuencia Alternativa		
Paso	**Acción**	**Respuesta**
S1.	Datos incorrectos.	Mostrar un mensaje que indique que los datos no son correctos, los posibles motivos, y que corrija el error. Ir a 2.

CASO DE USO #43	**Iniciar**
Objetivo en contexto	Iniciar la animación y resolución del problema escogido. La visualización se hará en función de la velocidad escogida. También se tendrán en cuanta la estrategia voraz en el esquema algorítmico voraz.
Entradas	No hay.
Precondiciones	Se han introducido los datos y el algoritmo está parado o es la primera que se inicia.
Salidas	Si no se introducen los datos se muestra un mensaje de error, si no, se

	anima el ejemplo escogido en este esquema algorítmico y se muestra el resultado obtenido hasta el momento.
Poscondición si éxito	Se ejecuta el ejemplo escogido.
Poscondición si fallo	No hay.
Actores	Usuario de la herramienta o usuario de desarrollo.

Secuencia normal

Paso	**Acción**	**Respuesta**
1.	Seleccionar la opción de "Iniciar" desde los menús o pulsando el botón correspondiente.	Comprobar que se han introducido los datos.
2a.	Si no se han introducido los datos.	Ir a S1.
2b.	Si se han introducido los datos.	Se verá la animación del algoritmo escogido desde el inicio del mismo.

Secuencia Alternativa

Paso	**Acción**	**Respuesta**
S1.	Los datos no se han introducido.	Mostrar un mensaje de error que indique que deben introducirse los datos de nuevo.

CASO DE USO #44	**Pausar**
Objetivo en contexto	Detiene la ejecución y animación del algoritmo momentáneamente hasta que se pulse el botón "Ejecutar".
Entradas	No hay.
Precondiciones	El algoritmo se ha iniciado y no está parado.
Salidas	Se detiene momentáneamente la animación del algoritmo y permanece, en el panel, el dibujo estático del algoritmo que se tenía en el momento en el que se pulsó la opción de "Pausar".
Poscondición si éxito	Se detiene la ejecución del algoritmo por el punto en el que estaba en el momento en el que se pulsó la opción de "Pausar".
Poscondición si fallo	No hay.
Actores	Usuario de la herramienta o usuario de desarrollo.

Secuencia normal		
Paso	**Acción**	**Respuesta**
1.	Seleccionar la opción de "Pausar" desde los menús o pulsando el botón correspondiente.	Se detiene la animación del algoritmo momentáneamente (hasta que se pulse el botón "Ejecutar"). Se mantiene en el panel el dibujo estático del algoritmo que se tenía en el momento en que se pulsó la opción de "Pausar".

CASO DE USO #45	**Ejecutar**
Objetivo en contexto	Proseguir la animación y ejecución del problema escogido. La visualización se hará en función de la velocidad escogida. También se tendrán en cuenta la estrategia voraz en el esquema algorítmico voraz.
Entradas	No hay.
Precondiciones	El algoritmo se ha pausado o ejecutado paso a paso con anterioridad y no está parado.
Salidas	Se anima el ejemplo escogido en este esquema algorítmico y se muestra el resultado obtenido hasta el momento.
Poscondición si éxito	Se ejecuta el ejemplo escogido desde el punto donde se detuvo al pausarse o desde el último punto que se ejecutó paso a paso.
Poscondición si fallo	No hay.
Actores	Usuario de la herramienta o usuario de desarrollo.

Secuencia normal		
Paso	**Acción**	**Respuesta**
1.	Seleccionar la opción de "Ejecutar" desde los menús o pulsando el botón correspondiente.	Se verá la animación del algoritmo escogido. Se ejecuta el ejemplo escogido desde el punto donde se detuvo al pausarse o desde el último punto que se ejecutó paso a paso.

CASO DE USO #46	**Ejecutar paso a paso**
Objetivo en contexto	Ejecutar un paso del algoritmo y visualizar la animación correspondiente a ese paso.
Entradas	No hay.
Precondiciones	Se han introducido los datos y el algoritmo está parado y no se ha iniciado,

	o bien el algoritmo se ha iniciado o pausado o ejecutado y no está parado.
Salidas	Si no se introducen los datos en caso de que el algoritmo esté parado y no se haya iniciado, se muestra un mensaje de error; si no, se muestra la animación del paso siguiente al que se encuentra el algoritmo.
Poscondición si éxito	Se ejecuta el ciclo siguiente al que se encuentra el algoritmo.
Poscondición si fallo	No hay.
Actores	Usuario de la herramienta o usuario de desarrollo.

Secuencia normal

Paso	Acción	Respuesta
1.	Seleccionar la opción de "Paso a paso" desde los menús o pulsando el botón correspondiente.	Comprobar que se han introducido los datos en caso de que el algoritmo esté parado y no iniciado.
2a.	Si no se han introducido los datos y el algoritmo está parado y no iniciado.	Ir a S1.
2b.	Si se han introducido los datos y el algoritmo está parado y no iniciado; o bien el algoritmo está iniciado o pausado o ejecutado y no parado.	Se ejecuta y anima el ciclo siguiente al que se encuentra el algoritmo.

Secuencia Alternativa

Paso	Acción	Respuesta
S1.	Datos no introducidos, algoritmo no iniciado y parado.	Mostrar un mensaje de error que indique que para ejecutar paso a paso desde el inicio se deberán introducir los datos necesarios para la resolución.

CASO DE USO #47	**Parar**
Objetivo en contexto	Detener completamente la ejecución del algoritmo y su animación.
Entradas	No hay.
Precondiciones	El algoritmo se ha iniciado, se ejecuta paso a paso, está pausado o se está ejecutando.

Salidas	En el panel se muestra el dibujo estático del algoritmo que se tenía en el momento en el que se pulsó la opción de “Parar”.
Poscondición si éxito	Se detiene la ejecución del algoritmo.
Poscondición si fallo	No hay.
Actores	Usuario de la herramienta o usuario de desarrollo.

Secuencia normal		
Paso	**Acción**	**Respuesta**
1.	Seleccionar la opción de “Parar” desde los menús o pulsando el botón correspondiente.	Se detiene la ejecución del algoritmo y en el panel se muestra el dibujo estático del algoritmo que se tenía en el momento en el que se pulsó la opción de “Parar”.

CASO DE USO #48	**Variar velocidad**
Objetivo en contexto	Variar la velocidad de ejecución y animación del algoritmo, para poder verlo más deprisa o más despacio. La velocidad puede variar en un rango comprendido entre 0 y 100.
Entradas	No hay.
Precondiciones	El algoritmo se ha iniciado, se ejecuta paso a paso, está pausado o se está ejecutando. La velocidad introducida por el usuario se encuentra dentro del rango fijado.
Salidas	La animación del algoritmo se realizará a la velocidad especificada por el usuario.
Poscondición si éxito	El hilo de la animación se dormirá más o menos tiempo en función de la velocidad seleccionada por el usuario.
Poscondición si fallo	No hay.
Actores	Usuario de la herramienta o usuario de desarrollo.

Secuencia normal		
Paso	**Acción**	**Respuesta**

1.	Seleccionar la opción de “Velocidad” desde el menú de herramientas o pulsando al botón correspondiente.	La ejecución y animación del algoritmo se efectuará más deprisa o más despacio en función de la velocidad seleccionada por el usuario. Cada vez que se pulse el botón “+” se incrementará la velocidad en 5. Cada vez que se pulse “-“ se minorará la velocidad en 5.

7.1. Visualización y Animación del Esquema algorítmico Voraz

CASO DE USO #49	**Visualización y animación del esquema algorítmico Voraz**	
Objetivo en contexto	Visualización y animación de la ejecución del algoritmo voraz que resuelve el problema de la mochila fraccionable o el algoritmo de Dijkstra.	
Entradas	No hay.	
Precondiciones	Se ha iniciado el algoritmo o se ha ejecutado paso a paso.	
Salidas	Visualización de la ejecución del problema de la mochila fraccionable o del algoritmo de Dijkstra y el resultado obtenido en la ejecución.	
Poscondición si éxito	Ejecución del algoritmo y obtención del resultado.	
Poscondición si fallo	No hay	
Actores	Usuario de la herramienta o usuario de desarrollo.	
Secuencia normal		
Paso	**Acción**	**Respuesta**
1	Insertar los datos de entrada requeridos. Seleccionar la opción de “Iniciar” o “Paso a paso” desde el menú de herramientas o pulsar a los botones iniciar o paso a paso.	Se ejecuta el algoritmo.

1a.	Si la aplicación seleccionada es el problema de la mochila fraccionable	Al iniciar la ejecución se mostrará la relación valor/peso existente entre cada objeto. En todo momento se podrá ver gráficamente la mochila construida hasta el momento con los objetos insertados en ella. Se visualizará la construcción de la mochila paso a paso, insertando los objetos seleccionados en cada etapa hasta alcanzar el tamaño máximo de la mochila. Además se visualizará una tabla, que va mostrando en cada etapa del proceso voraz, el valor y peso almacenado en la mochila, el candidato seleccionado y las porciones de cada objeto insertadas en ella. También se mostrará los datos de salida, esto es, las porciones finales de cada objeto que forman parte de la mochila. Las acciones que se podrán visualizar durante la ejecución del algoritmo son: - Pintar mochila vacía - Visualizar los candidatos, se mostrarán en el vector de entrada de color rosa. - Seleccionar el elemento más óptimo según la estrategia voraz elegida, se mostrarán en el vector de entrada de color turquesa. Además se insertará el objeto seleccionado en la mochila

1b.	Si la aplicación seleccionada es el algoritmo de Dijkstra	Se visualizará un grafo cuyos nodos representen las ciudades del recorrido y cuyas aristas determinen la distancia existente entre las mismas. Durante la ejecución del algoritmo se mostrará en cada etapa, con distintos colores, los nodos procesados hasta el momento y el nodo seleccionado (nodo cuya distancia al nodo origen es mínima). También se visualizará los nodos, no procesados hasta el momento, cuya distancia al nodo origen se vea modificada por el nuevo nodo seleccionado. Como datos de salida se mostrará sobre el grafo en un color más oscuro, los nodos y las aristas que formen parte de su camino mejor, y en una tabla se reflejará la distancia mínima que hay entre cualquier nodo y el nodo inicio. Las acciones que se podrán visualizar durante la ejecución del algoritmo son: - Pintar grafo en panel de dibujo - Mostrar nodo inicio en un tono oscuro - Asignar como distancia mínima entre cualquier nodo del grafo y el nodo inicio, la distancia existente entre ellos. Dicha operación se visualizará coloreando las celdas del vector de salida de color amarillo. - Seleccionar de los nodos no procesados el de menor distancia al nodo inicio. Las celdas del vector de salida correspondiente a dichos nodos se pintaran de color amarillo. - Visualizar el nodo seleccionado. Se coloreara de color rojo la celda del vector de salida correspondiente al nodo, y sobre el grafo se pintara, también de color rojo, dicho nodo y las aristas que forman parte de su camino mínimo. -

		- Recalcular la distancia mínima de los nodos no procesados, cuyas celdas se colorean en amarillo, a través del último nodo seleccionado. - Visualizar los nodos no procesados cuya distancia se modifica. La celda del vector de salida correspondiente al nodo se colorea en verde, y sobre el grafo se pintara el nodo y las aristas de su camino a partir del nodo seleccionado de color verde.

7.2. Visualización y Animación del Esquema algorítmico Programación Dinámica

CASO DE USO #50	**Visualización y animación del esquema algorítmico Programación Dinámica**	
Objetivo en contexto	Visualización y animación de la ejecución del algoritmo que resuelve el problema de la mochila 0-1	
Entradas	No hay.	
Precondiciones	Se ha iniciado el algoritmo o se ha ejecutado paso a paso.	
Salidas	Visualización de la ejecución del algoritmo de la mochila 0-1 y el resultado obtenido.	
Poscondición si éxito	Ejecución del algoritmo y obtención del resultado.	
Poscondición si fallo	No hay	
Actores	Usuario de la herramienta o usuario de desarrollo.	
Secuencia normal		
Paso	**Acción**	**Respuesta**

1	Insertar los datos de entrada requeridos. Seleccionar la opción de "Iniciar" o "Paso a paso" desde el menú de herramientas o pulsar a los botones iniciar o paso a paso.	Al resolver el problema de la mochila 0-1 podrá visualizarse la matriz con los objetos en las filas y el peso de la mochila en las columnas (desde cero hasta el peso máximo de la mochila). La matriz se irá rellenando con el valor de la mejor mochila, cuyo peso se corresponde con el indicado en la columna, utilizando los primeros i elementos, siendo i la fila. Cuando se haya rellenado toda la tabla se tendrá la solución de la mejor mochila con peso correspondiente a la capacidad máxima y utilizando todos los objetos, en la última celda de la matriz. Una vez construida toda la matriz, se reconstruirá la solución, mostrando los objetos que se han seleccionado en la mochila, el beneficio y el tamaño ocupado de la mochila, como datos de salida. Las acciones que se podrán visualizar durante la ejecución del algoritmo son: - Iniciar la construcción de la tabla - Rellenar la columna de peso cero con valor cero, mostrando las celdas implicadas de color gris. - Modificar el valor de una celda de la matriz: dicha celda se visualiza de color gris, coloreando también las celdas de la matriz y del vector de entrada, usadas para el cálculo del valor de la celda a modificar. Si se encuentra en el caso de obtener el máximo de dos posibles valores, se mostraran las celdas correspondientes a un posible valor de color azul claro, mientras que las otras celdas se mostraran de color azul oscuro. Las celdas utilizadas para obtener el valor se visualizaran de color rosa.

		- Reconstrucción de la solución: las celdas de color amarillo representarán las celdas utilizadas para decidir si el objeto i forma parte de la mochila. Las celdas de color verde de la matriz indicarán el objeto i que se esta procesando, los cuadros de texto se mostrarán de color verde cuando se modifique el valor del beneficio y capacidad ocupada de la mochila. En la lista se reflejará los objetos insertados.

7.3. Visualización y Animación del Esquema algorítmico Divide y vencerás

CASO DE USO #51	**Visualización y animación del esquema algorítmico Divide y Vencerás**	
Objetivo en contexto	Visualización y animación de la ejecución del algoritmo Quicksort o búsqueda binaria	
Entradas	No hay.	
Precondiciones	Se ha iniciado el algoritmo o se ha ejecutado paso a paso.	
Salidas	Visualización de la ejecución del algoritmo Quicksort o búsqueda binaria y el resultado obtenido.	
Poscondición si éxito	Ejecución del algoritmo y obtención del resultado.	
Poscondición si fallo	No hay	
Actores	Usuario de la herramienta o usuario de desarrollo.	
Secuencia normal		
Paso	**Acción**	**Respuesta**
1	Insertar los datos de entrada requeridos. Seleccionar la opción de “Iniciar” o “Paso a paso” desde el menú de herramientas o pulsar a los botones iniciar o paso a paso.	Se ejecuta el algoritmo.

1a.	Si la aplicación seleccionada es el algoritmo Quicksort	Durante la ejecución del algoritmo se seleccionará un elemento pivote (elemento situado en la primera posición del vector), se colocará los elementos más grandes que el elemento pivote a la derecha y los más pequeños a su izquierda. Una vez que se tenga esta semi-ordenación se procesará por separado, de la misma manera que la descrita anteriormente, la parte situada a la derecha del pivote y la parte izquierda. Este procesamiento se irá repitiendo hasta que cada parte del vector quede ordenada. Una vez que todas las partes estén ordenadas se tendrá el vector completo ordenado. Las acciones que se podrán visualizar durante la ejecución del algoritmo son: - Visualizar el trozo del vector que se está tratando - Mostrar el elemento pivote. - Visualizar los punteros izquierda y derecha. - Intercambiar los elementos, de manera que los de menor valor al pivote se sitúen en la parte izquierda y los mayores en la parte derecha. - Situar el elemento pivote en el medio del vector tratado.

1b.	Si la aplicación seleccionada es el algoritmo búsqueda binaria	Se introducen los datos de entrada en un vector y se reordenan mediante una llamada interna a Quicksort. En el panel de dibujo podrá verse el vector ordenado, sobre el cual se buscará la posición de un determinado elemento, mediante sucesivas divisiones de la parte que se procesa del vector. La parte que se está procesando del vector aparecerá en color mientras que la otra parte aparecerá en blanco y negro. Inicialmente la parte procesada será todo el vector. Se comprobará si el elemento a buscar se haya en la parte central de la parte procesada del vector. Si es así se habrá encontrado la solución, sino se comparará el elemento del medio con el elemento buscado, en función de lo cual se seleccionará la mitad donde seguir buscando. Si es más pequeño se cogerá como parte a procesar la mitad izquierda, sino se seleccionará la mitad derecha. Esto se hará hasta que no queden más elementos por procesar o bien hasta que se encuentre la solución. Las acciones que se podrán visualizar durante la ejecución del algoritmo son: - Visualizar el trozo del vector que se está tratando - Apuntar al elemento del medio, comparando el valor de dicho elemento con el buscado.

7.4. Visualización y Animación del Esquema algorítmico Ramificación y Poda

CASO DE USO #52	**Visualización y animación del esquema algorítmico de Ramificación y Poda**

Objetivo en contexto	Visualización y animación de la ejecución del algoritmo que resuelve el problema de la mochila 0-1 en el esquema algorítmico de ramificación y poda.	
Entradas	No hay.	
Precondiciones	Se ha iniciado el algoritmo o se ha ejecutado paso a paso.	
Salidas	Visualización y animación de los nodos que se generan del árbol (representan objetos que se introducen en la mochila) y de la cola de prioridad asociada a la construcción del árbol.	
Poscondición si éxito	Ejecución del algoritmo de la mochila	
Poscondición si fallo	No hay.	
Actores	Usuario de la herramienta o usuario de desarrollo.	
Secuencia normal		
Paso	**Acción**	**Respuesta**

1.	Insertar los datos de entrada requeridos. Seleccionar la opción de "Iniciar" o "Paso a paso" desde el menú de herramientas o pulsar a los botones iniciar o paso a paso.	Al resolver el problema de la mochila 0-1 podrá visualizarse el árbol que se va construyendo para buscar la solución y la cola de prioridad asociada al árbol. Cada nivel del árbol representará la adición de un objeto a la mochila. Cada arista representará si se inserta o no el objeto en la mochila. Cada nodo representa un llenado parcial de la mochila, que contendrá la etapa, peso acumulado, beneficio y beneficio óptimo. Cuando se llegue al peso máximo de la mochila se dejarán de generar nodos. Sólo se dibujarán los nodos del árbol que no se hayan podado. La cola de prioridad será de máximos puesto que es un problema de maximización. Los elementos de la cola serán los nodos con los llenados parciales de la mochila. La cola y el árbol evolucionan a la vez. En cada paso del algoritmo se seleccionará el nodo de la cola con mayor beneficio óptimo. De este nodo se crearán sus dos hijos (en el caso de que no se poden) y se insertarán en la cola. También se mostrará el resultado indicando qué objetos se han seleccionado y cuáles no para introducirse en la mochila. Las acciones que se podrán visualizar durante la ejecución del algoritmo son: - Visualizar el árbol creado hasta el momento y la cola de prioridad correspondiente. - Mostrar la raíz del árbol y el la cola de prioridad correspondiente. - Mostrar la construcción del árbol mediante la visualización de la creación de los hijos derecho e izquierdo y ver la cola de prioridad correspondiente. - Mostrar los nodos del árbol que se podan.

8. Documentación de los Esquemas Algorítmicos

CASO DE USO #53	**Mostrar código fuente del esquema algorítmico**
Objetivo en contexto	Se mostrará el código con el que se ha implementado el algoritmo seleccionado.
Entradas	Ninguna.
Precondiciones	El usuario se encuentra en la ventana principal del esquema algorítmico del que se quiere obtener el código fuente.
Salidas	Código fuente en Java del algoritmo seleccionado.
Poscondición si éxito	No hay.
Poscondición si fallo	No hay.
Actores	Usuario de la herramienta o usuario de desarrollo.

Secuencia normal

Paso	**Acción**	**Respuesta**
1.	Seleccionar desde el menú de herramientas la opción “Código”.	Mostrar en una ventana aparte una página de documentación HTML que muestre el código fuente del algoritmo seleccionado para ejecutar el esquema algorítmico.

CASO DE USO #54	**Mostrar coste del esquema algorítmico**
Objetivo en contexto	Visualizar una tabla que permita comparar el coste de ejecutar el algoritmo seleccionado en otros esquemas algorítmicos.
Entradas	Ninguna.
Precondiciones	El usuario se encuentra en la ventana principal del esquema algorítmico del que se quiere obtener el coste.
Salidas	Visualización de una tabla que permita mostrar el coste asintótico temporal y espacial del algoritmo seleccionado, en cada uno de los esquemas algorítmicos en los que se puede implementar el algoritmo en cuestión.

Poscondición si éxito	No hay
Poscondición si fallo	No hay
Actores	Usuario de la herramienta o usuario de desarrollo.
Secuencia normal	

Paso	**Acción**	**Respuesta**
1.	Seleccionar desde el menú de herramientas la opción “Coste”.	Mostrar en una ventana aparte una página de documentación HTML que muestre una tabla que permita comparar el coste asintótico temporal y espacial del algoritmo seleccionado, en cada uno de los esquemas algorítmicos en los que se puede implementar el algoritmo en cuestión.

CASO DE USO #55	**Mostrar ayuda adicional del esquema algorítmico**
Objetivo en contexto	Se dispondrá de una página de ayuda que contenga información general sobre el esquema algorítmico (esquema general del esquema algorítmico, características, etc).
Entradas	Ninguna.
Precondiciones	El usuario se encuentra en la ventana principal del esquema algorítmico del que se quiere obtener la ayuda adicional.
Salidas	Página de ayuda con información general sobre el esquema algorítmico.
Poscondición si éxito	No hay
Poscondición si fallo	No hay
Actores	Usuario de la herramienta o usuario de desarrollo.
Secuencia normal	

Paso	**Acción**	**Respuesta**

1.	Seleccionar desde el menú de herramientas la opción "Ayuda Adicional".	Mostrar en una ventana aparte una página de documentación HTML que muestre la información general sobre el esquema algorítmico.

9. Ayuda

CASO DE USO #56	**Índice de materias**
Objetivo en contexto	Mostrar pantalla de ayuda.
Entradas	No hay
Precondiciones	No hay
Salidas	Se muestra una pantalla de ayuda.
Poscondición si éxito	No hay
Poscondición si fallo	No hay
Actores	Usuario de la herramienta o usuario de desarrollo.

Secuencia normal		
Paso	**Acción**	**Respuesta**
1	Seleccionar la opción de "Ayuda → Índice" desde el menú de herramientas.	Se muestra una pantalla de ayuda en la cual a la izquierda aparece un índice de los temas de ayuda y a la derecha aparece el tema de ayuda concreto seleccionado. Si no se ha seleccionado ningún tema de ayuda en concreto aparece una documento de las características generales de la herramienta. Desde el índice se podrá acceder a toda la ayuda disponible en el sistema. Éste poseerá hipervínculos a los documentos de ayuda adicional de cada estructura de datos y esquema algorítmico.

CASO DE USO #57	**Manual de Usuario**

Objetivo en contexto	Mostrar pantalla de ayuda.
Entradas	No hay
Precondiciones	No hay
Salidas	Se muestra una pantalla de ayuda.
Poscondición si éxito	No hay
Poscondición si fallo	No hay
Actores	Usuario de la herramienta o usuario de desarrollo.

Secuencia normal		
Paso	**Acción**	**Respuesta**
1	Seleccionar la opción de "Ayuda → Manual de usuario" desde el menú de herramientas.	Se mostrará un manual de usuario que contendrá el soporte técnico necesario para el correcto funcionamiento del sistema. En él se describirá las posibles acciones que se pueden llevar a cabo, así como los resultados esperados que se han de obtener.

IMPLEMENTACIÓN: ASPECTOS TÉCNICOS

1. Motivación y alcance de la herramienta

El objetivo del proyecto es el desarrollo de una herramienta pedagógica, destinada a los alumnos universitarios pertenecientes a las facultades de informática, que estén cursando las asignaturas de estructuras de datos y esquemas algorítmicos. Esta herramienta se usará como método didáctico para facilitar el entendimiento, la práctica y la profundización en dichas asignaturas. Esta aplicación también resultará útil a los profesores que imparten dichas asignaturas.

Al tratarse de una aplicación gráfica, con dibujos y animaciones, resulta más sencillo comprender el funcionamiento de las estructuras de datos y el funcionamiento de los algoritmos.

Las estructuras de datos disponibles en la aplicación serán las siguientes:

- Pilas
- Colas
- Árboles (ABB, AVL)
- Colas de Prioridad

El usuario podrá consultar o modificar cada una de las estructuras anteriores, únicamente a través de las operaciones permitidas en las mismas. También podrá ver la especificación algebraica de la estructura seleccionada, las diversas formas de implementación de las estructuras y el coste espacial y temporal de cada implementación.

Los algoritmos disponibles en la aplicación se ajustan a los siguientes esquemas algorítmicos:

- Programación dinámica
- Divide y vencerás
- Devorador
- Ramificación y poda

Sobre cada uno de los esquemas algorítmicos se desarrollarán algoritmos concretos, de modo que el usuario pueda ver la utilidad de cada esquema algorítmico. En particular algunas de las funcionalidades ofrecidas al usuario serán la visualización del efecto de utilizar diversas estrategias voraces, la construcción de la matriz de programación dinámica, la selección de los caminos mínimos en el algoritmo de Dijkstra, el seguimiento de las llamadas recursivas realizadas en un algoritmo de divide y vencerás...

Hemos de destacar que a principios del mes de Marzo se hizo una presentación de la herramienta a los alumnos del grupo A de Estructuras de Datos y de la información de la Ingeniería Informática.

La presentación y prueba de la herramienta por parte de los alumnos es uno de los objetivos finales del proyecto, al tratarse la herramienta de una aplicación pedagógica.

2. Plataformas y tecnologías utilizadas

2.1 Java versus C++

La plataforma escogida para el desarrollo de la herramienta ha sido Java. No se ha escogido C++ ya que Java parecía más apropiado para el desarrollo de una herramienta gráfica y en general se considera que la programación en Java es más limpia que la de C++.

2.2 Java Development Kit (JDK)

A la hora de la selección de la JDK usada para compilar el proyecto se optó por la elección de la maquina virtual JDK1.3 frente a la JDK1.4 en base a que la última no permitía visualizar las páginas web incluidas en el proyecto de manera correcta. En particular, no permitía visualizar las CSS internas, ni mostraba el estilo de letra seleccionado, ni otras opciones de configuración del documento.

2.3 Plataforma Gráfica

Una de las primeras decisiones que se tomaron a la hora de realizar el presente proyecto fue la elección de la plataforma que se iba a utilizar para visualizar la parte gráfica del proyecto.

Se encontraron las siguientes alternativas:

- Java 2D
- Java 3D
- OpenGL

Se descartó la integración de OpenGL sobre Java ya que la integración de los mismos aún no está madura.

Aunque Java 3D ofrecía bastantes facilidades a la hora de realizar simulaciones, se optó por la elección de Java 2D, ya que el proyecto no requería de excesiva potencia gráfica. En los gráficos requeridos para representar el funcionamiento de las EDs y esquemas algorítmicos, la incorporación de la tercera dimensión no aportaba una ayuda adicional al usuario, así que se optó por el uso de gráficos planos.

2.4 Clase Graphics versus Graphics 2D

Para representar gráficos 2D en Java existen dos clases: Graphics 2D y Graphics.

Se optó por la selección de la clase Graphics 2D frente a la clase Graphics ya que además de poseer todas las características de la clase Graphics incluye mejoras como la de un control más sofisticado de la geometría y nuevas opciones como la posibilidad de manejar transformaciones de coordenadas, manejo del color, etc...

2.5 Java 2D

En el apéndice A se han recopilado algunas de las funcionalidades que ofrece Java 2D y que han sido utilizadas para el desarrollo de la parte gráfica de la aplicación.

3. Diseño de la herramienta

Cuando se diseña una aplicación, se ha de procurar ante todo que sea lo más modular posible, de tal forma que la introducción de nuevo conocimiento sea lo más fácil posible para el programador. Este ha sido nuestro afán a la hora de diseñar la herramienta, elaborar una aplicación que pudiera completarse con nuevas estructuras de datos y algoritmos sin que supusiera demasiado esfuerzo. Para ello se han realizado tres módulos principales, la interfaz, el paquete gráfico y el paquete de implementación, de tal forma que la implementación no se mezcle nunca con la parte gráfica.

En la figura 1 se representa un esquema del diseño que se utilizado para desarrollar la herramienta. A continuación se explicarán sus características principales.

El usuario se comunica con la herramienta a través de la interfaz, la cual posee una parte interactiva y una parte a la que denominaremos *JPanel*, donde podrá visualizar la animación de las distintas EDs y algoritmos. La parte de visualización mostrará los efectos de la interacción del usuario con la parte interactiva mediante dibujos y animaciones.

La parte interactiva, en función de las operaciones aplicadas por el usuario, llamará a la parte de implementación de las EDs o de los algoritmos, las cuales devolverán:

- En el caso de trabajar con una ED: la ED modificada y un vector de acciones que se usará para la posterior representación gráfica y,
- En el caso de trabajar con un algoritmo, el resultado de la ejecución del problema con el que se está trabajando y un vector de acciones para la representación gráfica posterior.

A su vez la parte interactiva llama al panel de dibujo, el cual actúa a modo de fachada, es decir, se encarga de la comunicación con la parte gráfica, de tal forma que cuando la interfaz desee pintar deberá pedírselo al panel de dibujo, el cual mostrará los resultados obtenidos de su llamada a la correspondiente parte gráfica sobre el *JPanel.*

En la parte gráfica se han desarrollado tanto gráficos, como animaciones de las distintas EDs y algoritmos. El movimiento de las animaciones que tienen lugar en la parte gráfica son controladas mediante hebras.

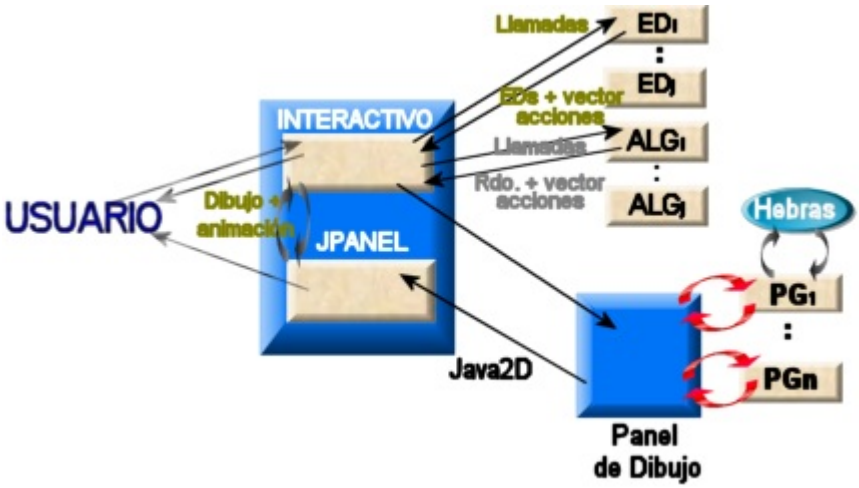


Figura 1: Diseño de la herramienta

Este diseño supone que cada parte es responsable de una tarea, de tal forma que la parte gráfica sólo dibujará y animará, y la parte de implementación únicamente efectuará las operaciones de la correspondiente ED o resolverá un determinado problema algorítmico. Esto implica que la parte de implementación no sabe dibujar y que la parte gráfica no sabe cómo está implementada la ED o algoritmo.

Con posterioridad se explicará más en detalle el diseño seguido, centrándonos en cada uno de los módulos desarrollados.

4. Parte interactiva y paneles gráficos

4.1 Parte interactiva

La interfaz desarrollada en la aplicación trata de ser intuitiva, clara y fácil de manejar. Las ventanas que forman parte de ella, se han descrito con detalle en el documento de Especificación de Requisitos y en el Manual de Usuario.

Mediante la interacción con nuestro sistema, el usuario podrá aprender a conocer el comportamiento de las EDs, las acciones factibles según el estado de la misma, las diferentes implementaciones, así como diversa documentación referente a las estructuras. También podrá seguir paso a paso la ejecución de un algoritmo y las partes implicadas en la resolución.

El usuario tendrá la oportunidad de trabajar directamente sobre las EDs y algoritmos, insertando los datos y seleccionando las acciones a realizar. O bien, podrá ver el funcionamiento de la aplicación, sin requerir ningún conocimiento previo, seleccionando una opción para realizar una simulación.

4.2 JPanel

La clase JPanel de Java permite la construcción de los paneles gráficos que contendrán los paneles de dibujo, funciones o datos del problema. En la sección 5 se describe con más detalle.

4.3 Índice

La herramienta, además de la visualización del comportamiento de las estructuras de datos y algoritmos, nos ofrece una serie de páginas web con diferente información acerca de las estructuras de datos, tales como su especificación algebraica, código, coste, e información adicional. En los algoritmos se podrá acceder a su código, coste e información adicional. A dichas páginas se puede acceder por medio de los menús.

También se dispondrá de un manual de usuario y de un índice de materias mediante el cual el usuario podrá buscar la toda la información disponible sobre las distintas EDs y algoritmos. Este índice esta formado por una parte en la que se encuentra las distintas materias en forma de árbol. Al pulsar sobre cada una de las materias contenidas se podrá ver el contenido de las mismas. Si el usuario lo desea puede volver a consultar la documentación que ha examinado con anterioridad. Para ello sólo ha de pulsar sobre el botón “Atrás”, el cual se sitúa en la parte superior izquierda.

Para la implementación del índice se ha usado un *JTree*, esta componente tiene un oyente para captar las pulsaciones del ratón sobre los distintos nodos del árbol tales como *addTreeSelectionListener*, pero debido a que esta función no funciona correctamente cuando los nodos antecesores del nodo que queremos seleccionar no se han pulsado, se ha optado por usar la clase *MouseAdapter* y su oyente *mousePressed* que controla las pulsaciones dadas por el usuario a través del ratón.

Cada nodo hoja del *JTree* contiene uno de los posibles temas que puede consultar el usuario. Al pulsar el usuario sobre una determinada temática, aparecerá en el *JEditorPane* que aparece a su derecha la información seleccionada, la cual es una página HTML sobre la que se puede navegar.

Otras de las características que posee el índice es la posibilidad de volver a visualizar de nuevo las páginas ya visitadas mediante la pulsación sobre el botón “Atrás”. Esta funcionalidad

se ha implementado usando una pila: cuando se consulta un tema nuevo se apila en la pila, de forma que cuando se pulse atrás, desapilando se pueda volver a la página ya consultada.

4.4 Documentación de la aplicación

Toda la documentación de la herramienta se ha realizado usando páginas HTML. Para la visualización de estas páginas HTML se ha usado un *JEditorPane*.

En un principio las páginas web creadas se hicieron con textos flash y CSS externas. Se tuvo que desechar esta opción ya que Java no era capaz de visionar en los *JEditorPane* estas características, así como otras como el texto justificado, los márgenes y algún que otro carácter. Se optó como solución por crear CSS internas al documento HTML y cambiar la alineación justificada del texto por alineación a la izquierda.

5. Panel de dibujo

La clase que se encarga de la comunicación entre la interfaz y la parte gráfica es el panel de dibujo. La interfaz ha de comunicarle al panel de dibujo lo que quiere que se pinte, para ello ha de pasarle a este panel un conjunto de acciones y una ED o un resultado dependiendo de si se trata de pintar una acción sobre una ED o del efecto de un algoritmo.

La clase Panel de dibujo extiende a *JPanel* y que necesita implementar su método *paintComponent*. Esta necesidad se produce por dos causas:

- Porque se requiere que se repinte el *JPanel* que está contenido en el *JFrame* de la interfaz y la manera de hacerlo es llamando a este método *paintComponent*
- Porque este método *paintComponent* es el que proporciona el objeto de tipo *Graphics* que se necesita para pintar, por tanto el método *paintComponent* será el que se encargue de llamar a todos los métodos de la parte gráfica.

El método *paintComponent* es llamado automáticamente por el gestor de eventos de la interfaz cuando se abre por primera vez la ventana que contiene el *JPanel* y cuando sucede alguno de los eventos de la ventana tales como minimizar, maximizar, etc. Se puede forzar el repintado desde la interfaz llamando a *super.repaint()*. A continuación explicaremos más detenidamente todo lo referente a este método.

5.1 PaintComponent

Siempre que pintemos lo haremos encima de un *JPanel*. La clase *JPanel* posee un método denominado *paintComponent* que recibe como parámetro un objeto *Graphics*, el cual se usará para pintar.

Cada vez que se necesita volver a dibujar la pantalla, el gestor de eventos ejecuta en todos los componentes el método *paintComponent(Graphics g).*

Para asegurarse de que la superclase realiza el trabajo de pintar la ventana se llama dentro del *paintComponent(Graphics g)* a *super.paintComponent(g)*.

Para forzar a hacer un redibujado del panel, se puede usar el método *repaint()*, el cual hará un redibujado tan pronto como sea posible; esto es, el sistema hará el *repaint* en el momento que tenga posibilidad de hacerlo. El sistema pone el trabajo en la cola de trabajos a realizar y lo hará cuando le llegue su turno ya que *repaint* tiene menos prioridad que otros.

Cuando se fuerza un repintado desde la interfaz llamando a *repaint*, el repintado se realiza siempre cuando finaliza el método desde el que se ha llamado a *repaint* debido a la concurrencia de las componentes de la interfaz. Esto hace que no se puedan hacer animaciones que incluyan varias acciones llamando a distintos métodos de la interfaz, veamos como ejemplo el método que realiza la simulación de la pila. Para hacer esta simulación habría varias posibilidades:

1. Llamar a las distintas operaciones desde la interfaz, esto es, si la operación es crear llamar al método crear de la interfaz, si la operación es apilar llamar al método nuevo elemento, etc., donde cada uno de estos métodos contiene una llamada a *repaint*. ¿Qué es lo que ocurre? Debido a la concurrencia de las componentes de la interfaz, todos los *repaint* de los distintos métodos llamados se ejecutan al final del método simulación de tal forma que sólo podemos visualizar la última operación aunque pongamos un hilo que se duerma cierto tiempo entre operación y operación.
2. Hacer que sea el *paintComponent* el que controle el pintado de las distintas acciones, es decir, la parte de implementación está separada de la parte gráfica de modo que la implementación efectúa las operaciones correspondientes y rellena un vector con los datos necesarios para el dibujado. El panel de dibujo lo que recorre este vector de acciones en el *paintComponent()* llamando en cada vuelta al pintar de la acción correspondiente. Esta opción sería factible si en la simulación de la pila no estuvieran las animaciones de apilar y de desapilar, ya que cada vez que se realiza la animación de una de estas operaciones se están haciendo llamadas al *paintComponent* para que se visualice cada uno de los pasos de la animación. Esto provoca un bucle infinito de llamadas a *repaint*.
3. Separar igual que antes la parte de implementación de la parte gráfica, de forma que la implementación haga las tareas que le correspondan y rellene un vector con los datos necesarios para el dibujado. A diferencia con la opción anterior, la parte gráfica es la que se ocupa de recorrer el vector de acciones y de pintar cada acción. En este caso el panel de dibujo es un simple comunicador.

Las opciones 1 y 2 se han implementado y se han probado, llegando a la conclusión de que no funcionan correctamente o que con ellas no obtenemos los resultados deseados, por ello se ha escogido la última opción, la cual no crea problemas en las llamadas a *repaint* y es la que se utilizará a la hora de realizar las distintas simulaciones.

6. Estructuras de datos y Algoritmos: Implementación y Parte Gráfica

6.1 Implementación

6.1.1 Clases propias versus clases genéricas

Cuando se necesita la implementación de estructuras de datos, primero debe obtenerse información acerca de las implementaciones existentes de dichas estructuras.

JBuilder tiene disponibles la implementación de las pilas (clase *Stack*), tablas hash (*HashMap, HashSet*), árboles (*TreeMap, TreeSet*) y otras estructuras.

Sin embargo, no hemos usado las implementaciones existentes sino las que hemos desarrollado nosotras. Se escogió esta opción porque necesitamos introducir métodos muy específicos para pintar las estructuras.

Las estructuras de datos utilizadas son genéricas, es decir, que pueden contener elementos de cualquier tipo. Esto se consigue en Java haciendo que los elementos de las estructuras sean de tipo *Object*. Una vez establecido el tipo de la ED, todos los elementos de la estructura deben ser de ese tipo.

6.1.2 Iteradores para recorrer las Estructuras de Datos

A pesar de que las estructuras de datos son tipos protegidos y encapsulados, necesitamos acceder a su estructura interna para poder pintarlas. Los iteradores son una forma segura de acceder a las estructuras de datos.

Los iteradores permiten acceder y recorrer estructuras de datos de forma más eficiente y sin modificarlas. Esto permite disponer de los elementos de la estructura sin destruirla.

Pondremos un ejemplo para comprender las ventajas de los iteradores y el modo en el que los hemos usado en nuestra aplicación.

En nuestra herramienta necesitamos acceder a estructuras de datos para poder obtener información de las mismas que nos permita dibujarlas.

Si se trata de una ED y dicho acceso a la estructura se hiciera mediante un bucle, tendríamos que hacer una copia de la pila original para no perder su información, ya que para obtener todos sus elementos es necesario sacar los elementos de la pila, con lo que la pila original se destruiría. No parece apropiado tener que realizar la copia en este caso.

Para evitar esto podemos usar los iteradores. El uso del iterador nos permite recorrer la estructura y acceder a sus elementos sin sacarlos de la estructura y por tanto, sin destruirla. El iterador permite ahorrar el tiempo y el espacio que supone hacer la copia de la estructura de la que se quiere obtener la información. Además, nos aseguramos de que nuestra ED no se habrá modificado tras el recorrido del iterador.

Los iteradores también pueden resultar útiles para hacer copias de las estructuras de datos de forma más eficiente.

Esto mismo también sucede con las colas, las colas de prioridad..., con todas las estructuras de datos que no tengan acceso directo a cada uno de sus elementos.

Hay varias posibilidades para incluir iteradores a una estructura.

1. Pueden implementarse los iteradores en la clase de la estructura de datos.
2. Pueden usarse los iteradores que ofrece Java.

Cuando somos nosotros los que implementamos el iterador, no se ha usado la opción de implementar los iteradores en una clase distinta de la estructura, ya que hace falta acceder a la estructura y sus atributos no pueden ser públicos.

Para evitar esto, podría ponerse la clase iterador dentro de la clase de la estructura. Eso sería similar a implementar los métodos del iterador directamente en la clase de la estructura.

Esta última es la opción que se ha elegido en el código que se muestra en las páginas web en las que hemos implementado nuestros propios iteradores.

En la implementación de la aplicación hemos aprovechado la implementación de los iteradores de Java para recorrer las estructuras (opción 2). Para poder aplicar un iterador sobre una estructura, la clase de la ED debe heredar de *AbstractList*. Además, la ED debe implementar los métodos *get (int i)* de acceso al elemento i-ésimo de la estructura y el método *size ()* que nos indica el tamaño de la estructura.

El iterador se crea e inicializa usando el método *iterator* de la ED, que nos devolverá un objeto del tipo *Iterator*. Se puede recorrer la estructura con un bucle y haciendo uso de los métodos *hasNext* y *next* del iterador. El primer método indica si el iterador tiene más elementos que recorrer en la estructura y el segundo devuelve, en el caso de que quedaran elementos por recorrer, el elemento correspondiente del recorrido y avanza al siguiente elemento.

La ventaja de los iteradores de Java es que tienen implementado el control de la concurrencia, aunque en nuestro caso no es necesario ya que al recorrer las estructuras no las modificamos.

6.1.3 Vectores de Acciones Simples versus Compuestas

Estructuras de datos

Como se ha dicho con anterioridad, cuando la interfaz llama a la parte de implementación, recibe de la misma un conjunto de acciones. Dependiendo de la ED estas acciones pueden ser simples o compuestas. Entendemos por acción simple a aquella acción que no requiere una composición de animaciones, por ejemplo una acción simple sería la inserción de un elemento el la cola; mientras que nos encontraremos ante una acción compuesta si tenemos una animación que consta de varias animaciones independientes, por ejemplo la inserción en un árbol AVL requiere una acción compuesta ya que además de insertar hay que equilibrar, por tanto además de pasarle las acciones necesarias para la inserción le pasamos las acciones necesarias para el equilibrio, que a su vez son acciones compuestas.

En todas las EDs tendremos que crear el panel de dibujo que servirá de comunicación entre la interfaz y la parte gráfica en la función crear de la interfaz, que será donde le pasemos al panel de dibujo la ED que estemos usando. El vector de acciones se le pasará al panel de dibujo únicamente cuando sea necesario; por ejemplo, cuando se esté efectuando una operación consultora tal como preguntar si la pila es vacía, no será necesario.

La información de los vectores de acciones se rellenará en la parte de implementación del algoritmo o en la interfaz en el caso de las EDs.

Pila

Únicamente se le pasará al panel de dibujo acciones en las operaciones de apilar y desapilar, las cuales serán acciones simples. Al apilar únicamente se indicará que la acción es apilar, y al desapilar se indicará que la acción es desapilar y cuál es el elemento que se ha suprimido.

Cola

Al panel de dibujo únicamente se le pasarán acciones en las operaciones de insertar y eliminar, las cuales serán acciones simples. Al insertar se indicará que la acción es insertar, y al eliminar que la acción es eliminar y se indicará cuál es el elemento que se ha suprimido.

Cola de prioridad

En la cola de prioridad distinguiremos si nos encontramos en la vista usuario o en el resto de las vistas ya que el comportamiento es diferente. Si nos encontramos en la vista usuario pasaremos al panel de dibujo un vector con los elementos ordenados según como se han introducido en la cola de prioridad.

Al insertar deberemos pasarle al panel de dibujo una acción compuesta. Esta acción estará formada por la indicación del tipo de operación que se está efectuando (inserción) junto con los montículos intermedios y con los elementos que se intercambian al flotar.

Si nos encontramos ante una eliminación habremos de pasar al panel de dibujo una acción compuesta que contendrá que la operación es una eliminación, el elemento que se elimina y los montículos intermedios que se generan al hundir, además de las posiciones de elementos que se intercambian durante este proceso.

Árbol Binario de Búsqueda

Tanto en los árboles binarios de búsqueda como en los AVL tendremos de pasarle al Panel de dibujo información sobre operaciones consultoras de recorridos, de elementos de un nivel y del hijo izquierdo, derecho y raíz además de la información sobre las operaciones de inserción y eliminación.

- En las operaciones de inserción se tendrá que pasar al panel de dibujo la operación que se está realizando, la posición donde se insertará el nuevo elemento.
- En las operaciones de eliminación se pasará al panel de dibujo una acción compuesta que contendrá la siguiente información. Si la operación es una eliminación pasaremos al panel de dibujo la acción compuesta formada por el elemento que se ha eliminado, su posición, el árbol anterior a la eliminación y la posición el elemento que se coloca en el lugar del elemento eliminado.
- En las operaciones de consulta de hijo izquierdo, derecho y raíz se pasará una acción simple que indicará el tipo de operación que se está realizando.

- En las operaciones de recorridos pasaremos además de la operación que se está realizando, los elementos que resultan de la realización del recorrido junto con su posición en el árbol.
- En la operación de consulta de los elementos de un nivel pasaremos además de la operación que se está realizando, la posición de los elementos pertenecientes a ese nivel.

Árbol Equilibrado AVL

En la inserción se le pasará al panel de dibujo una acción compuesta. Esta acción compuesta estará formada por el tipo de operación que se está realizando (inserción), la posición donde se inserta el elemento, el árbol antes del desequilibrio y un vector que contendrá la información necesaria para realizar la rotación que corresponda. Este vector estará por parejas que contendrán el subárbol que se equilibra y las distintas acciones que usaremos para pintar el equilibrado: si la rotación es izquierda o derecha, por dónde se empieza a equilibrar y los árboles intermedios que se generan.

Si la operación es una eliminación, pasaremos al panel de dibujo la acción compuesta formada por el elemento que se ha eliminado, su posición, el árbol anterior a la eliminación, la posición el elemento que se coloca en el lugar del elemento eliminado y un vector con las acciones atómicas necesarias para representar el equilibrado.

En las operaciones consultoras de recorridos, de hijo izquierdo, derecho y raíz y de los elementos de un nivel pasaremos la misma información que en el árbol BB.

Esquemas Algorítmicos

Durante la ejecución de los algoritmos, se almacenará en un vector las acciones simples que se deberán realizar al visualizar el mismo. Dicho vector se enviará a la parte gráfica correspondiente. Cada acción representará un paso de ejecución del algoritmo.

Algoritmo voraz que resuelve el problema de la mochila

Las acciones que se podrán llevar a cabo en cada paso son:

- Pintar mochila vacía
- Visualizar los candidatos
- Visualizar el elemento insertado en la mochila, éste será el elemento óptimo según la estrategia voraz elegida.

Algoritmo de Dijkstra

Las acciones que se podrán llevar a cabo en cada paso consistirán en:

- Pintar el grafo en el panel de dibujo
- Mostrar el nodo inicio
- Asignar como distancia mínima entre cualquier nodo del grafo y el nodo inicio, la distancia existente entre ellos.
- Seleccionar de los nodos no procesados el de menor distancia al nodo inicio.
- Visualizar el nodo seleccionado y las aristas que forman parte de su camino mínimo.
- Recalcular la distancia mínima de los nodos no procesados, a través del último nodo seleccionado.
- Visualizar los nodos no procesados cuya distancia se modifica. En el vector de salida se reflejará la nueva distancia mínima.

Algorítmico de Programación dinámica que resuelve el problema de la mochila

Las acciones que se podrán llevar a cabo en cada paso consistirán en:

- Iniciar la construcción de la tabla
- Asignar las casillas de la columna cero al valor cero.
- Determinar el valor de cada celda, visualizando las celdas a partir de las cuales se obtiene el nuevo valor.
- Mostrar la reconstrucción de la solución así como los objetos seleccionados, el valor y espacio final.

Algoritmo de Búsqueda Binaria

Las acciones que se podrán llevar a cabo en cada paso consistirán en:

- Visualizar el trozo del vector que se está tratando.
- Apuntar al elemento del medio, comparando el valor de dicho elemento con el buscado.

Algoritmo Quicksort

Las acciones que se podrán llevar a cabo en cada paso consistirán en:

- Visualizar el trozo del vector que se está tratando.
- Mostrar el elemento pivote.
- Visualizar los punteros izquierda y derecha.
- Intercambiar los elementos, de manera que los de menor valor al pivote se sitúen en la parte izquierda y los mayores en la parte derecha.
- Situar el elemento pivote en el medio del vector tratado.

Algoritmo de Ramificación y Poda que resuelve el problema de la mochila

Las acciones que se podrán llevar a cabo en cada paso consistirán en:

- Visualizar el árbol creado hasta el momento y la cola de prioridad correspondiente.
- Mostrar la raíz del árbol y el la cola de prioridad correspondiente.
- Mostrar la construcción del árbol mediante la visualización de la creación de los hijos derecho e izquierdo y ver la cola de prioridad correspondiente.
- Mostrar los nodos del árbol que se podan.

6.2 Parte Grafica

6.2.1 Animaciones

A la hora de realizar animaciones se han distinguido dos tipos distintos en función del número de acciones que las conformen:

- Animaciones de acciones simples
- Animaciones de acciones compuestas

Animaciones de acciones simples

Se ha considerado como acción simple a aquellas acciones que implican un movimiento, por ejemplo el movimiento que provoca la inserción de un elemento en la pila.

Las animaciones de acciones simples se realizan con hebras, de modo que cada objeto sabe cómo moverse y es la hebra la que controla el movimiento y la velocidad. El movimiento se consigue mediante las llamadas que hace el método *run* de la hebra al método mover, que es el que controla la dirección del movimiento (actualiza la posición del objeto). El método mover tras la realización del movimiento, llamará a su vez a *repaint*, lográndose de ese modo el encadenamiento de movimientos.

El movimiento se iniciará cuando llame al método *start* del hilo y parará cuando se cumpla una determinada condición de terminación, por ejemplo llegar al punto (x,y), que hará que muera el hilo.

Nótese que esta forma de animación mediante hilos permite el movimiento de distintos objetos a distinta velocidad.

Entre otras decisiones de diseño con respecto a este apartado se optó por crear un hilo únicamente cuando había animación, ya que por ejemplo, si se tuviera una pila

y se estuviera consultando si es vacía no se requeriría tener el hilo activo. Se evita estar consumiendo recursos del procesador.

Otra decisión de diseño fue que no se podía consultar si había terminado un hilo con un booleano ya que como *repaint* es un hilo interno, podía ser que se llamara a *repaint* antes que al hilo y el hilo no supiera que había terminado y siguiera activo. En esta situación se tendrían dos hilos con el mismo nombre y fallaría la ejecución. Como se ha dicho, el control de la finalización del hilo se controla viendo si se cumple o no una determinada condición.

Animaciones de acciones compuestas

Se ha considerado como acción compuesta a aquella acción que está formada por encadenamiento de acciones, entre las que se pueden dar acciones simples (movimientos). Las animaciones de acciones compuestas se usan tanto en las simulaciones como en la visualización de algoritmos.

Las animaciones de acciones compuestas se realizan aprovechando las múltiples llamadas que se realizan al hilo del *repaint*.

Una animación de este tipo consistiría por ejemplo, en la composición de las siguientes acciones en el caso de la pila

Crear pila
Apilar 3
Apilar 4
Consultar si es vacía
Desapilar
....

Cada acción individual tiene un intervalo de tiempo en el que se realizaría y tras el paso de ese intervalo de tiempo se pasa a la visualización de la siguiente acción. En el caso de que la acción a realizar sea una acción simple se debe esperar, además del tiempo que le corresponde a cada acción, a que se termine la acción simple (el movimiento).

La animación de acciones compuestas termina cuando se realicen todas las acciones que la componen.

7. Pasos para extender la herramienta

La herramienta está desarrollada de tal forma que desarrolladores posteriores puedan extenderla de forma sencilla, sin que esta tarea demande grandes requerimientos de tiempo y esfuerzo.

La forma de introducir una nueva estructura de datos en la herramienta es la siguiente:

- Implementar las siguientes partes:
  1. Implementar la ED
  2. Realizar la representación gráfica de la estructura y las animaciones correspondientes
  3. Realizar la interfaz de la ED
- Una vez que tenemos estas partes, hemos de juntarlas. Para ello, cada funcionalidad de la interfaz deberá llamar a las operaciones correspondientes de la estructura implementada y, si la operación requiere pintar sobre el panel, se deberán pasar las acciones necesarias para el dibujado al panel de dibujo y hacer un *super.repaint()*. Como ya se ha contado en apartados anteriores el *repaint* hará una llamada al *paintComponent* del panel de dibujo, el cual será el encargado de llamar a la correspondiente operación de la parte gráfica.

  El panel de dibujo se creará en el método que desarrolla la operación crear de la interfaz.
- En el panel de dibujo deberemos hacer las siguientes acciones:
  - En el constructor se ha de crear el objeto de la estructura gráfica correspondiente, de forma que se le pase el objeto que representa la implementación de la estructura, ya que a partir de él vamos a dibujar la estructura.
  - En el *paintComponent* se debe crear un caso para la ED y llamar en él al método que se encarga de dibujar la ED correspondiente.
  - Si la operación a dibujar requiere la animación de una acción simple se ha de crear en el *paintComponent* el hilo que controle la animación.

La forma de introducir un nuevo algoritmo en la herramienta es como se muestra a continuación. Cabe decir que cada algoritmo tiene una forma particular de realizarse.

Implementar las siguientes partes:

1. Implementar el algoritmo en una clase. La clase contiene un método que implementa el algoritmo. Dicho método debe devolver relleno el vector de acciones con la información necesaria para pintar el algoritmo.
2. Realizar la representación gráfica deseada del algoritmo y las animaciones correspondientes.
3. Realizar la interfaz del algoritmo. La interfaz es la encargada de realizar la petición de los datos de entrada del algoritmo, de mostrar los datos de salida.

Una vez que tenemos estas partes, hemos de juntarlas. Para ello:

- En el método *iniciar* de la clase *Interfaz* debe crearse un caso para el nuevo algoritmo.

  En él debe crearse el objeto de la clase *PanelDibujo*. Además, debe crearse un objeto de la clase en la que se haya implementado el algoritmo y llamar con este objeto al método que

ejecuta el algoritmo. Después se le debe pasar al panel de dibujo el vector de acciones devuelto por la llamada realizada.

Al final de los métodos que se encargan de iniciar y dar un paso en un algoritmo se llama a *super.repaint()*.

En algunos algoritmos podría ser necesario modificar en la interfaz los métodos encargados de ejecutar el algoritmo, pararlo, dar un paso, aumentar y disminuir la velocidad.

En la clase *PanelDibujo* debe realizarse lo siguiente:

- Crear un atributo privado en la clase *PanelDibujo* de tipo *Vector* que representará el vector de acciones necesarias para pintar el algoritmo.
- En el constructor de la clase referente a los algoritmos debe crearse un caso para el nuevo algoritmo en el que se asigne el vector de acciones para pintar el algoritmo (que se recibirá como parámetro del constructor) al vector de acciones de la clase. Además, se creará el objeto de la clase gráfica que dibujará el algoritmo.
- En el método *paintComponent* de esta clase debe crearse un caso para el nuevo algoritmo. En algunos algoritmos es necesario que en la primera llamada se le pase el vector de acciones al objeto gráfico del algoritmo.

Se llama al método del objeto gráfico del algoritmo y se le pasan como argumentos el objeto de la clase *Graphics* que es el que nos permitirá dibujar, el vector de acciones, el estado de ejecución del algoritmo (iniciado, parado, paso a paso, velocidad) y el panel gráfico sobre el que se dibujará.

En la clase del paquete gráfico que se encarga de pintar el algoritmo:

- Debe tenerse en cuenta el estado de ejecución del algoritmo para pintar lo que corresponda.
- Deben pintarse las diferentas acciones del algoritmo. Esto se puede controlar con las variables *numAcciones* y *contEsperaAccion* que representan respectivamente el contador del número de acciones que se han de pintar y el tiempo que se ha de pintar cada acción.
- Después de pintar cada acción se hará un *super.repaint()*.
- En este método de pintado debe crearse además el hilo de las animaciones, en el caso de que las hubiera.

Para volver a visualizar la acción anterior de la ejecución de un algoritmo, habría que restar 1 a la variable *numacciones* y proceder de la misma manera .

8. Trabajo futuro

La herramienta podría ampliarse en un futuro añadiéndose las páginas web que no se hayan incluído en los algoritmos.

Como tarea futura queda añadir estructuras de datos y algoritmos más complejos. Como estructuras de datos se podrían añadir por ejemplo árboles roji-negros, árboles B y grafos. En la parte de algoritmos se podría incluir el esquema algorítmico de vuelta atrás, algoritmos que resuelvan el problema del viajante, etc. También en la parte de algoritmos podría hacerse que se pudiera volver a ver el paso anterior de la ejecución del algoritmo. En el apartado de “pasos para extender la herramienta” se explica cómo hacer estas extensiones con facilidad.

Además, dado el propósito de este proyecto, sería de gran interés realizar pruebas masivas de la herramienta por parte de los alumnos y profesores para probar la utilidad de la herramienta. Sería interesante que la aplicación la probaran tanto alumnos que aún no han cursado la asignatura de estructura de datos, alumnos que han cursado esa asignatura pero no han cursado esquemas algorítmicos y alumnos que han cursado ambas asignaturas para obtener opiniones desde diferentes puntos de vista. También sería interesante la aportación de las pruebas de los profesores de la herramienta.

1. Introducción

Para el análisis y diseño de nuestro sistema, nos hemos basado en UML, el Lenguaje Unificado de Modelado definido por Booch, Jacobson y Rumbaugh. UML es una notación estándar que soporta el modelo de objetos, y que sirve para visualizar, especificar, construir y documentar los elementos de un sistema software.

Existen diferentes herramientas CASE que ayudan al modelado de los sistemas de acuerdo con esta notación. En nuestro caso, hemos utilizado la herramienta Racional Rose y hemos elaborado los siguientes tipos de diagramas: de casos de uso, de actividades, de clases, de interacción (secuencia) y de despliegue.

2. Diagrama de Casos de Uso

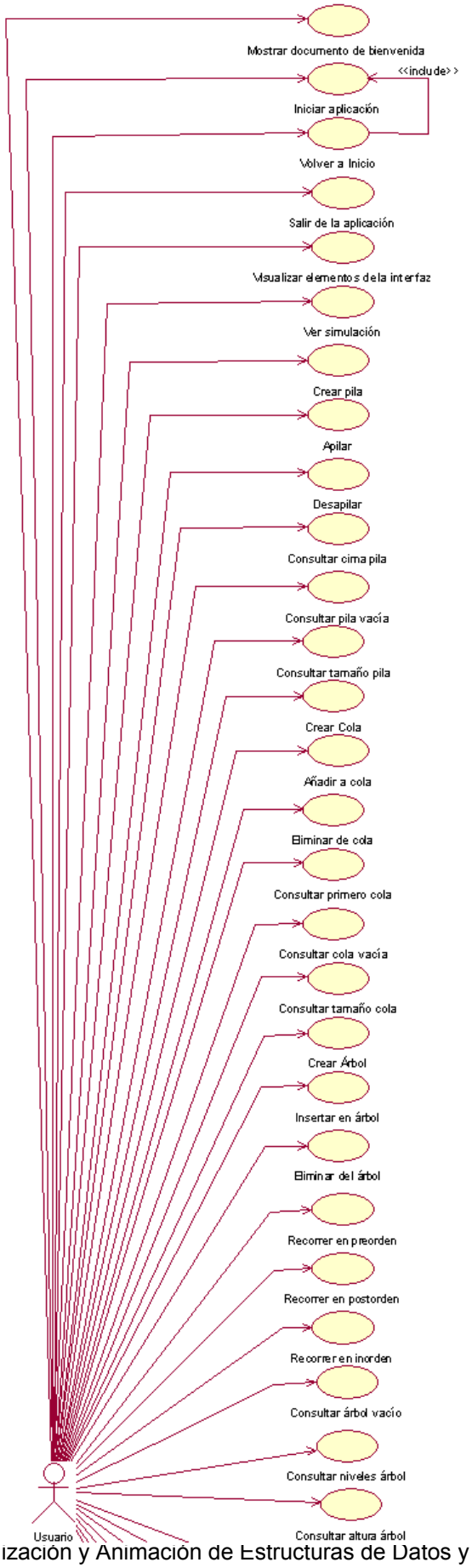
Un caso de uso es un patrón de comportamiento que debe existir en el sistema. Cada caso de uso es una secuencia de transacciones realizadas por un actor. Para la realización del diagrama de casos de uso, se examinan los actores y se determinan sus necesidades.

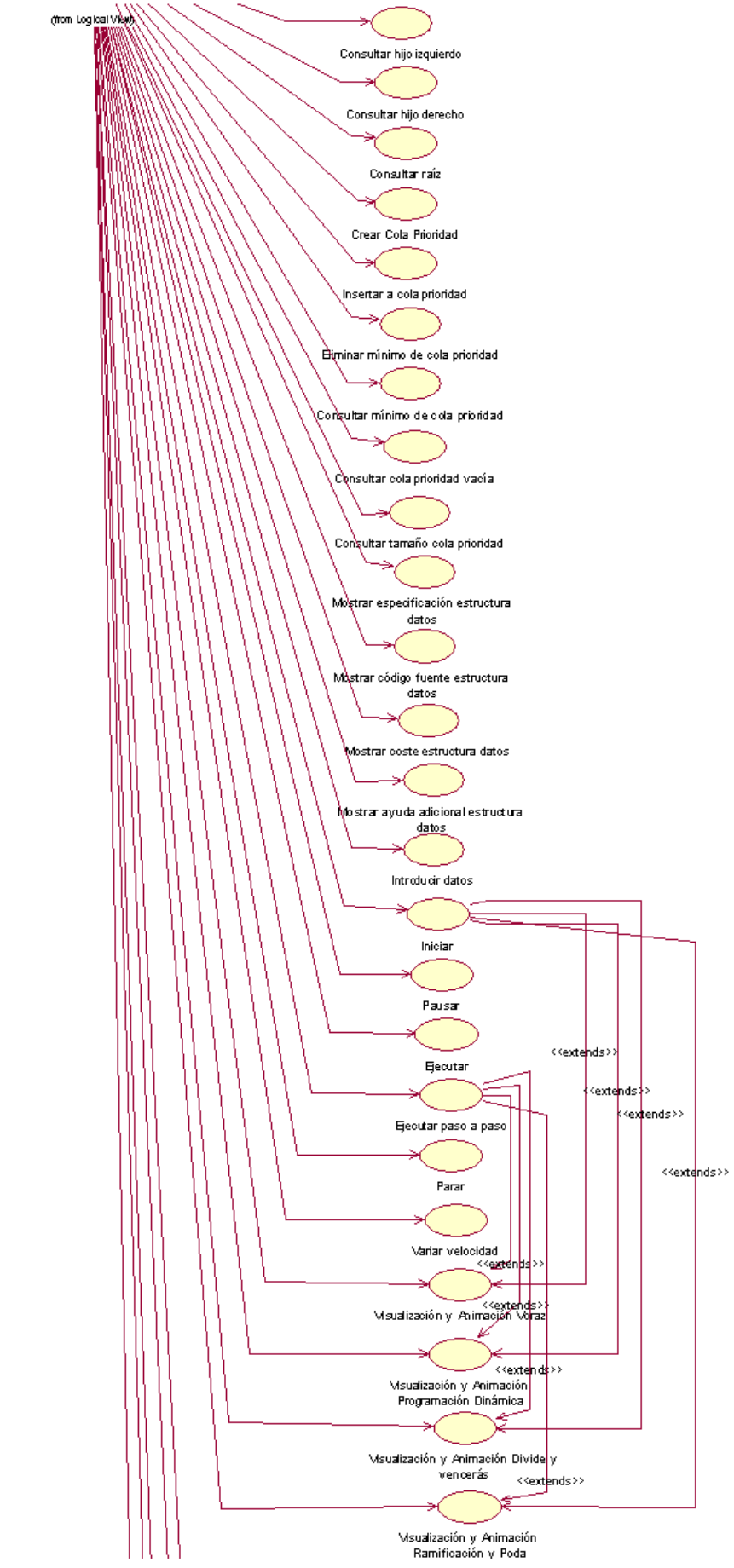
En nuestro sistema existe tan sólo un actor, el usuario. Los comportamientos o funciones que puede desempeñar en el sistema son las siguientes:

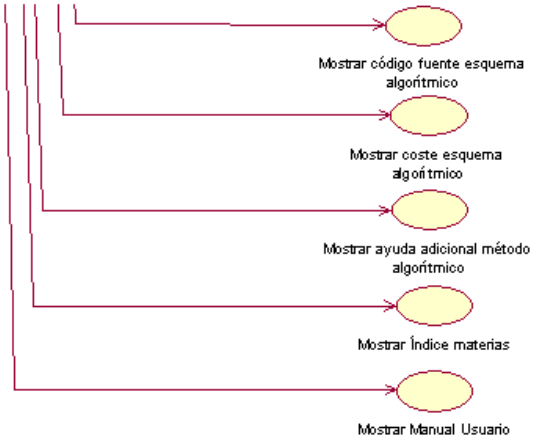
- Mostrar documento de bienvenida
- Iniciar aplicación
- Volver a Inicio
- Salir de la aplicación
- Visualizar elementos de la interfaz
- Ver simulación
- Crear pila
- Apilar
- Desapilar
- Consultar cima pila
- Consultar pila vacía
- Consultar tamaño pila
- Crear Cola
- Añadir a cola
- Eliminar de cola

- Consultar primero cola
- Consultar cola vacía
- Consultar tamaño cola
- Crear Árbol
- Insertar en árbol
- Eliminar del árbol
- Recorrer en preorden
- Recorrer en postorden
- Recorrer en inorden
- Consultar árbol vacío
- Consultar niveles árbol
- Consultar altura árbol
- Consultar hijo izquierdo
- Consultar hijo derecho
- Consultar raíz
- Crear Cola de Prioridad
- Insertar a cola de prioridad
- Eliminar de cola de prioridad
- Consultar mínimo de cola de prioridad
- Consultar cola de prioridad vacía
- Consultar tamaño cola de prioridad
- Mostrar especificación estructura de datos
- Mostrar código fuente estructura de datos
- Mostrar coste estructura de datos
- Mostrar ayuda adicional estructura de datos
- Introducir datos
- Iniciar
- Pausar
- Ejecutar
- Ejecutar paso a paso
- Parar
- Variar velocidad
- Visualización y animación Voraz
- Visualización y animación Programación Dinámica
- Visualización y animación Divide y Vencerás
- Visualización y animación Ramificación y Poda
- Mostrar código fuente del esquema algorítmico

- Mostrar coste esquema algorítmico
- Mostrar ayuda adicional esquema algorítmico
- Mostrar Índice de materias
- Mostrar Manual de Usuario


Mostrar documento de bienvenida
<<include>>
Iniciar aplicación
Volver a Inicio
Salir de la aplicación
Visualizar elementos de la interfaz
Ver simulación
Crear pila
Apilar
Desapilar
Consultar cima pila
Consultar pila vacía
Consultar tamaño pila
Crear Cola
Añadir a cola
Eliminar de cola
Consultar primero cola
Consultar cola vacía
Consultar tamaño cola
Crear Árbol
Insertar en árbol
Eliminar del árbol
Recorrer en preorden
Recorrer en postorden
Recorrer en inorden
Consultar árbol vacío
Consultar niveles árbol
Consultar altura árbol
Usuario


(from Logical View)
Consultar hijo izquierdo
Consultar hijo derecho
Consultar raíz
Crear Cola Prioridad
Insertar a cola prioridad
Eliminar mínimo de cola prioridad
Consultar mínimo de cola prioridad
Consultar cola prioridad vacía
Consultar tamaño cola prioridad
Mostrar especificación estructura datos
Mostrar código fuente estructura datos
Mostrar coste estructura datos
Mostrar ayuda adicional estructura datos
Introducir datos
Iniciar
Pausar
Ejecutar
Ejecutar paso a paso
Parar
Variar velocidad
Visualización y Animación Voraz
Visualización y Animación Programación Dinámica
Visualización y Animación Divide y vencerás
Visualización y Animación Ramificación y Poda
<<extends>>



3. Diagramas de Actividades

Un diagrama de actividades muestra el flujo de actividades a desarrollar para un determinado caso de uso. Los diagramas de actividades especifican los casos de uso sin definir el comportamiento de la clase ni las interrelaciones entre objetos.

Como muestra, se presentan los diagramas de actividades más representativos:

- Consultar si la pila está vacía
- Apilar
- Desafilar
- Visualización y animación del algoritmo Quicksort

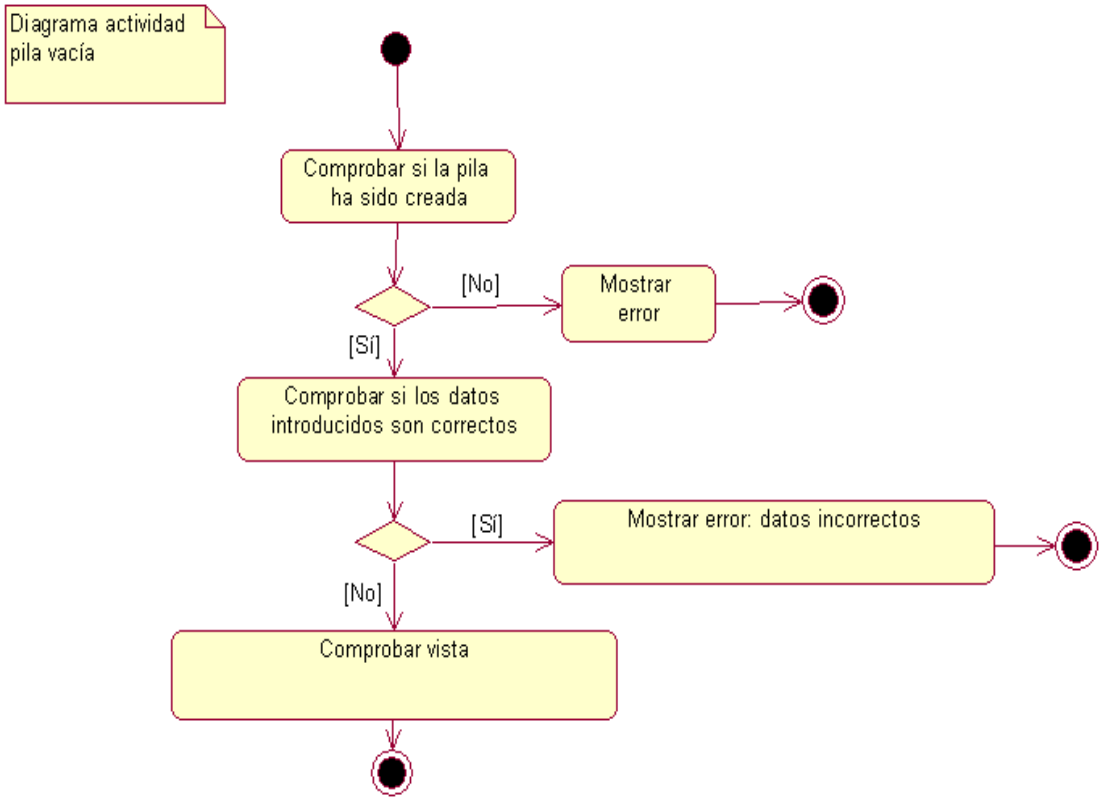


Diagrama actividad
pila vacía
Comprobar si la pila
ha sido creada
[No]
Mostrar
error
[Sí]
Comprobar si los datos
introducidos son correctos
[Sí]
Mostrar error: datos incorrectos
[No]
Comprobar vista

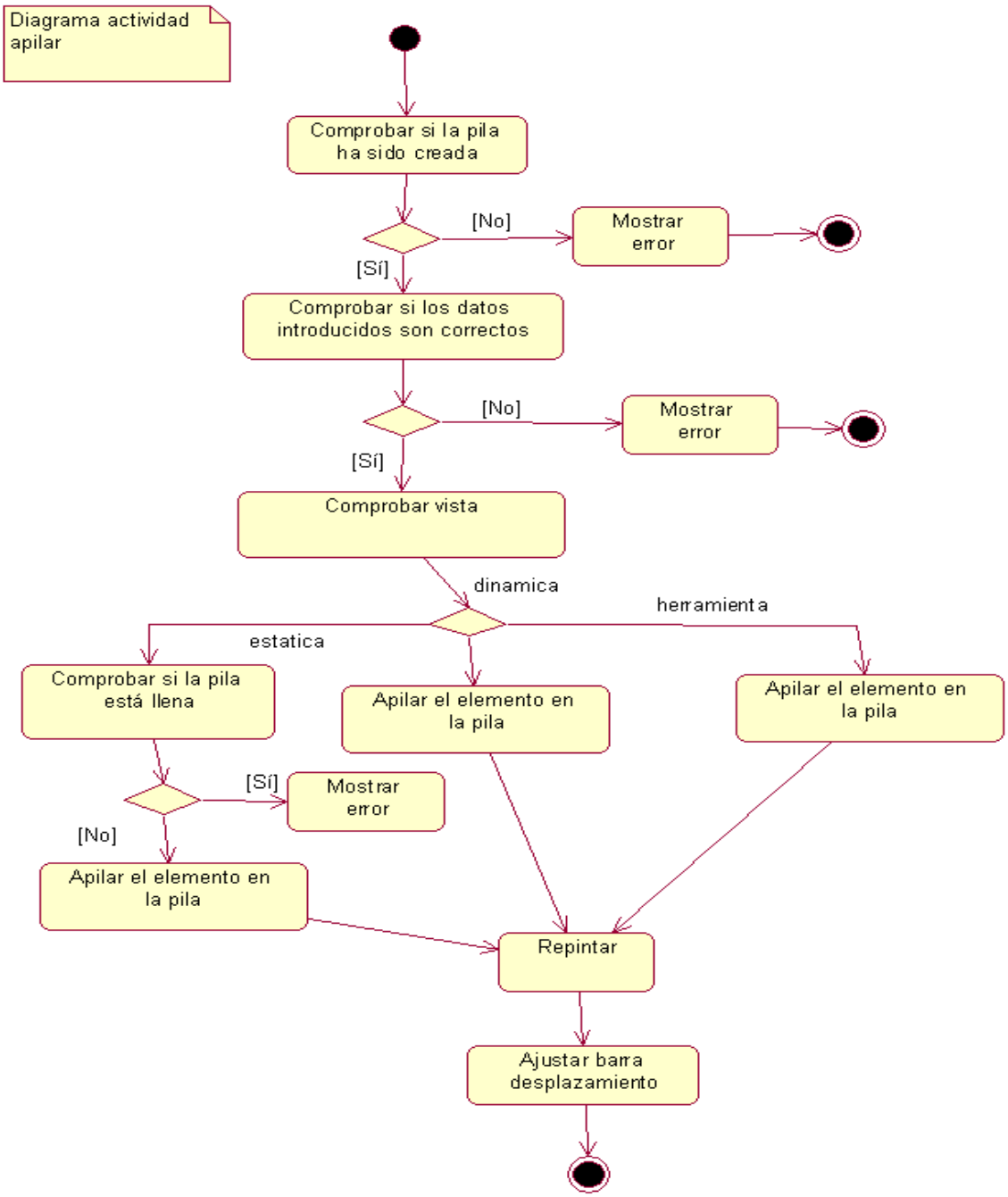


Diagrama actividad apilar
Comprobar si la pila ha sido creada
[No]
Mostrar error
[Sí]
Comprobar si los datos introducidos son correctos
[No]
Mostrar error
[Sí]
Comprobar vista
dinamica
herramienta
estatica
Comprobar si la pila está llena
Apilar el elemento en la pila
Apilar el elemento en la pila
[Sí]
Mostrar error
[No]
Apilar el elemento en la pila
Repintar
Ajustar barra desplazamiento

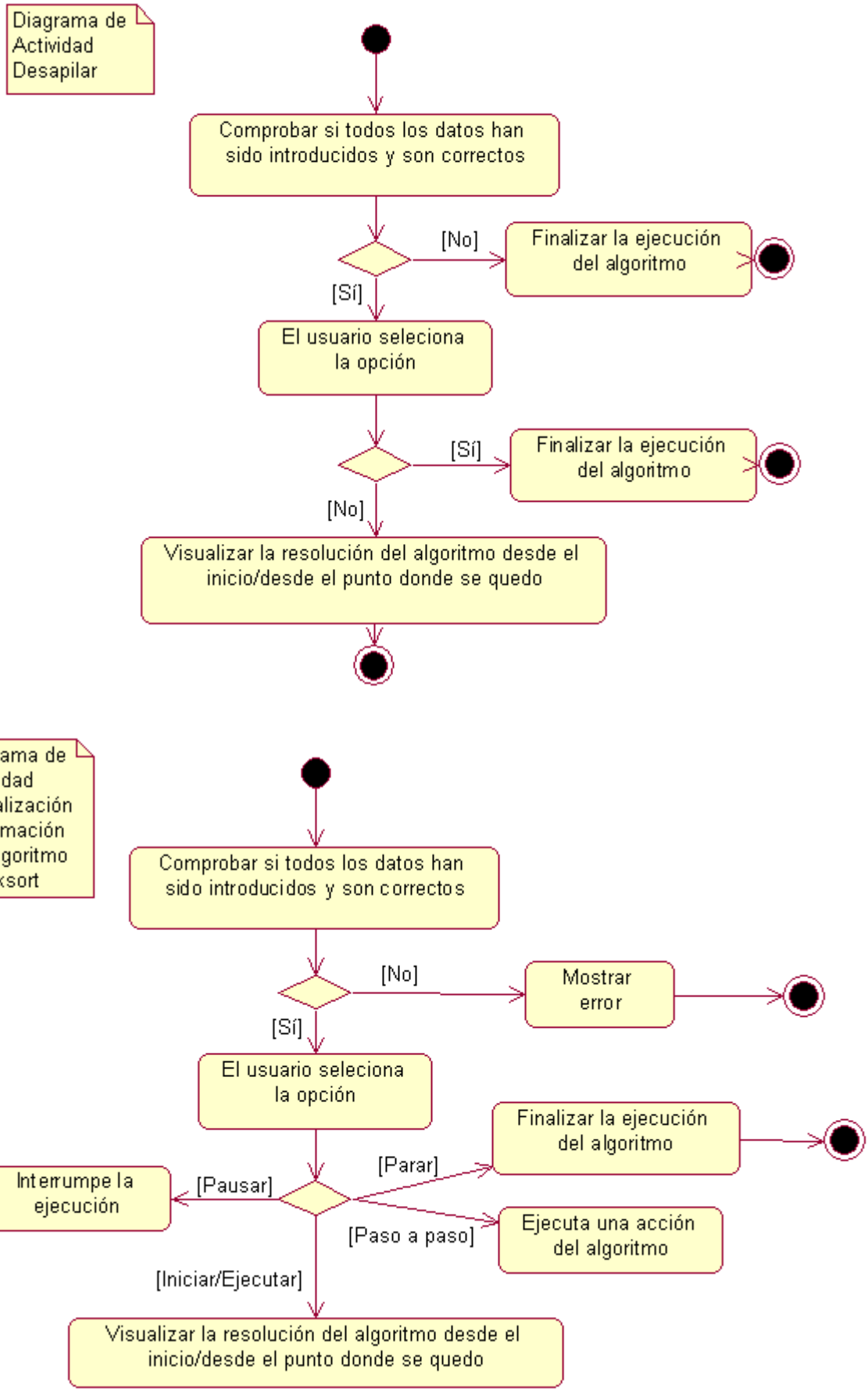
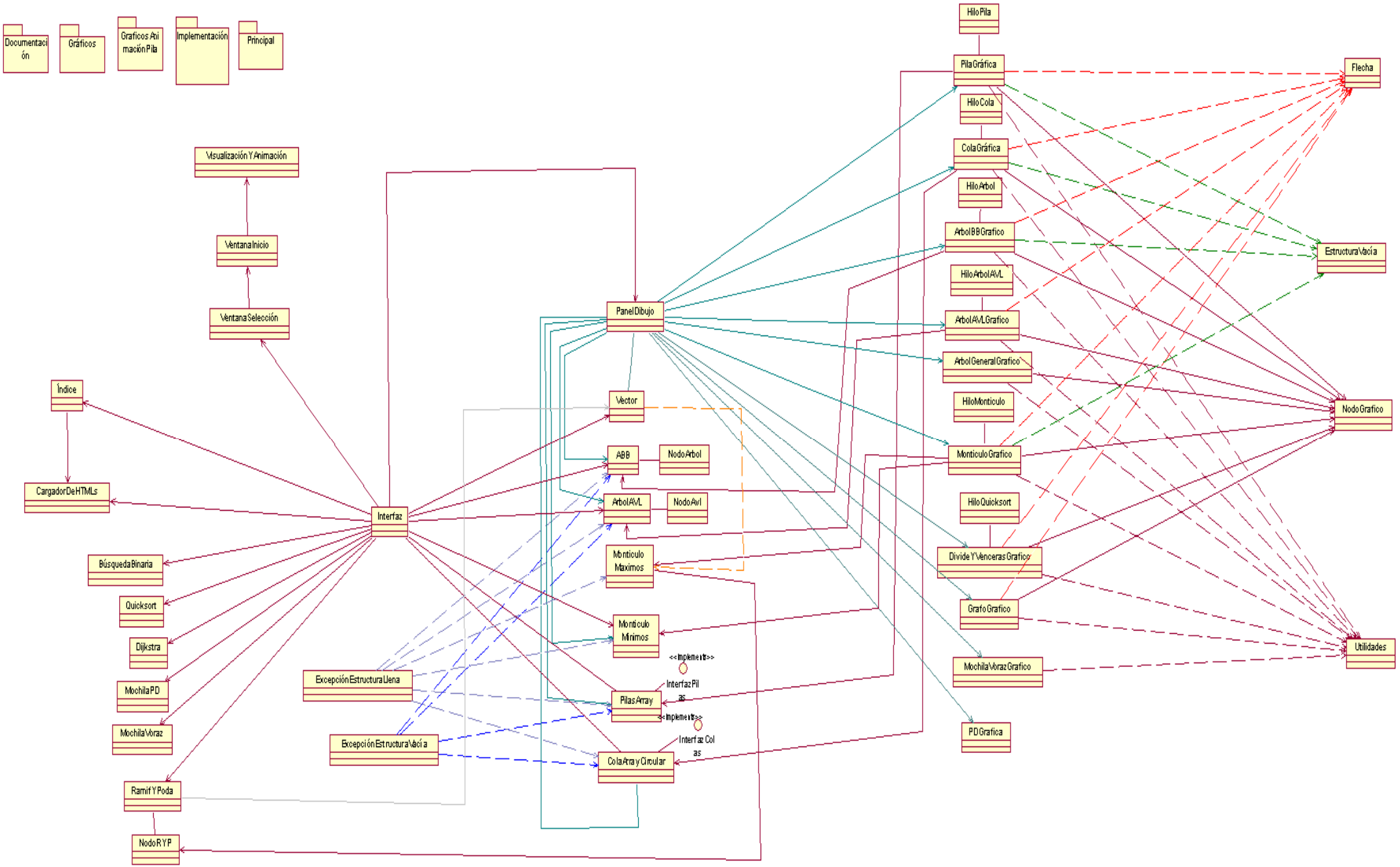


Diagrama de Actividad Desapilar
Comprobar si todos los datos han sido introducidos y son correctos
[No]
Finalizar la ejecución del algoritmo
[Sí]
El usuario seleciona la opción
[Sí]
Finalizar la ejecución del algoritmo
[No]
Visualizar la resolución del algoritmo desde el inicio/desde el punto donde se quedo
Diagrama de Actividad Visualización y Animación del algoritmo Quicksort
Comprobar si todos los datos han sido introducidos y son correctos
[No]
Mostrar error
[Sí]
El usuario seleciona la opción
Finalizar la ejecución del algoritmo
[Parar]
Interrumpe la ejecución
[Pausar]
[Paso a paso]
Ejecuta una acción del algoritmo
[Iniciar/Ejecutar]
Visualizar la resolución del algoritmo desde el inicio/desde el punto donde se quedo

4. Diagrama de Clases

Un diagrama de clases muestra las clases y sus relaciones desde el punto de vista lógico. En un diagrama de clases se representan los siguientes elementos de modelado UML:

- Clase, su estructura y comportamiento.
- Asociación, agregación, dependencia y relaciones de herencia.
- Indicadores de navegación.


Documentación
Gráficos
Graficos Animación Pila
Implementación
Principal
Visualización Y Animación
VentanaInicio
VentanaSelección
Índice
CargadorDeHTMLs
Interfaz
BúsquedaBinaria
Quicksort
Dijkstra
MochilaPD
MochilaVoraz
RamifYPoda
NodoRYP
PanelDibujo
Vector
ABB
NodoArbol
ArbolAVL
NodoAvl
MonticuloMaximos
MonticuloMinimos
ExcepciónEstructuraLlena
ExcepciónEstructuraVacía
<<Implements>>
InterfazPilas
PilasArray
<<Implements>>
InterfazColas
ColaArrayCircular
HiloPila
PilaGráfica
HiloCola
ColaGráfica
HiloArbol
ArbolBBGrafico
HiloArbolAVL
ArbolAVLGrafico
ArbolGeneralGrafico
HiloMonticulo
MonticuloGrafico
HiloQuicksort
DivideYVenceras Grafico
GrafoGrafico
MochilaVorazGrafico
PDGrafica
Flecha
EstructuraVacía
NodoGrafico
Utilidades

5. Diagramas de Interacción

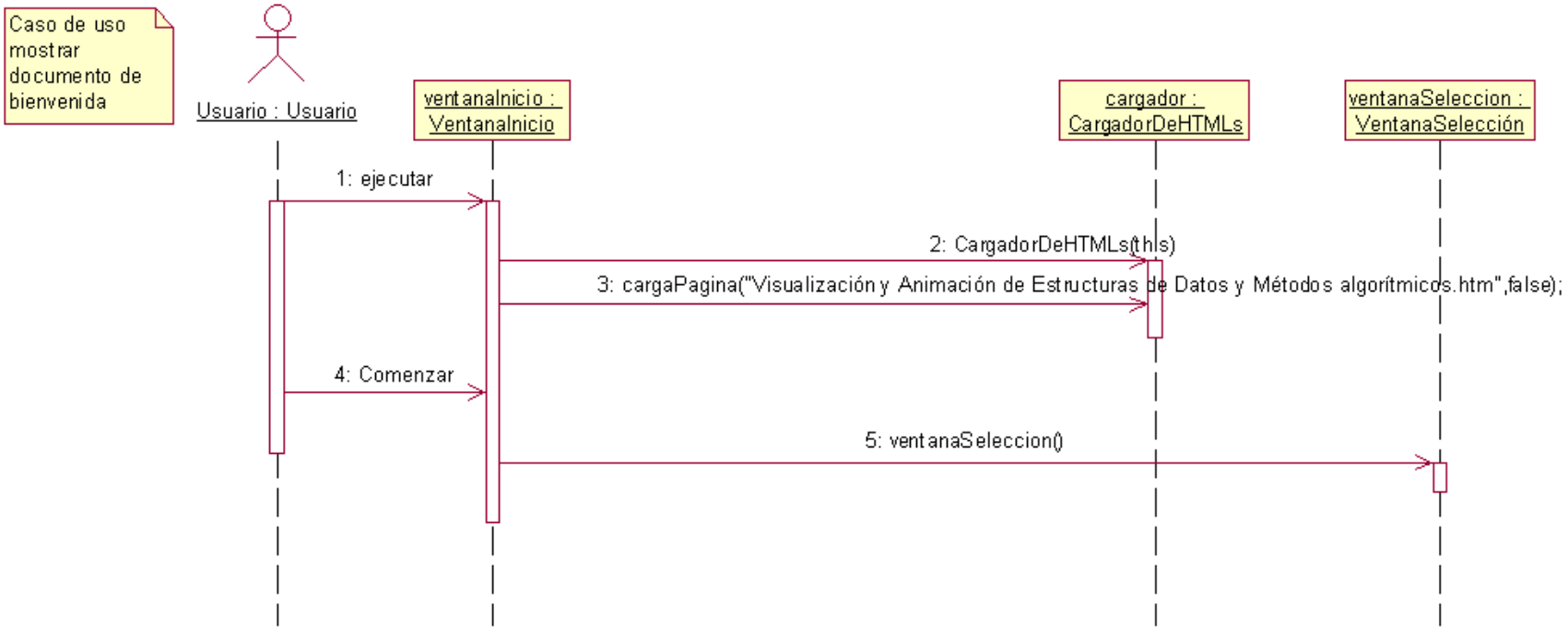
Los diagramas de interacción describen la realización de los casos de uso como interacciones entre objetos. Existen dos tipos de diagramas de interacción: los diagramas de secuencia y los diagramas de colaboración.

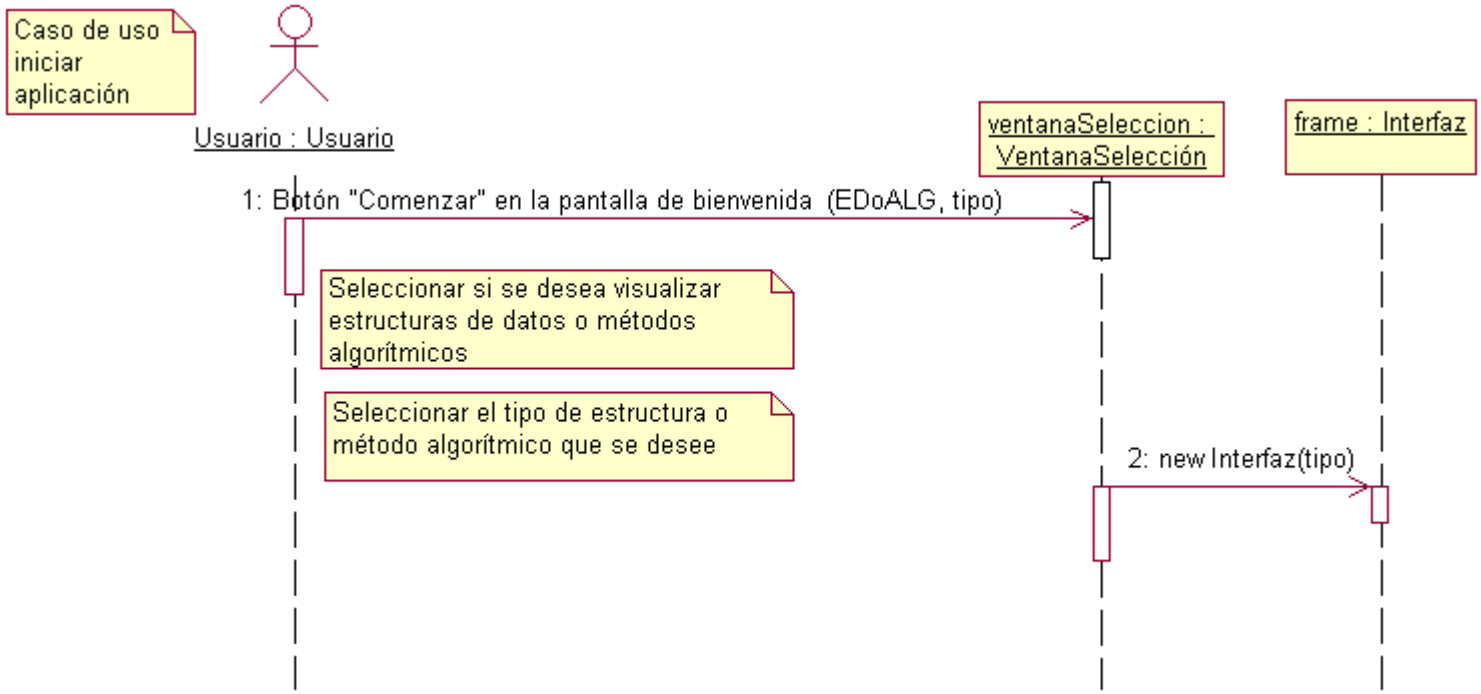
5.1 Diagramas de Secuencia

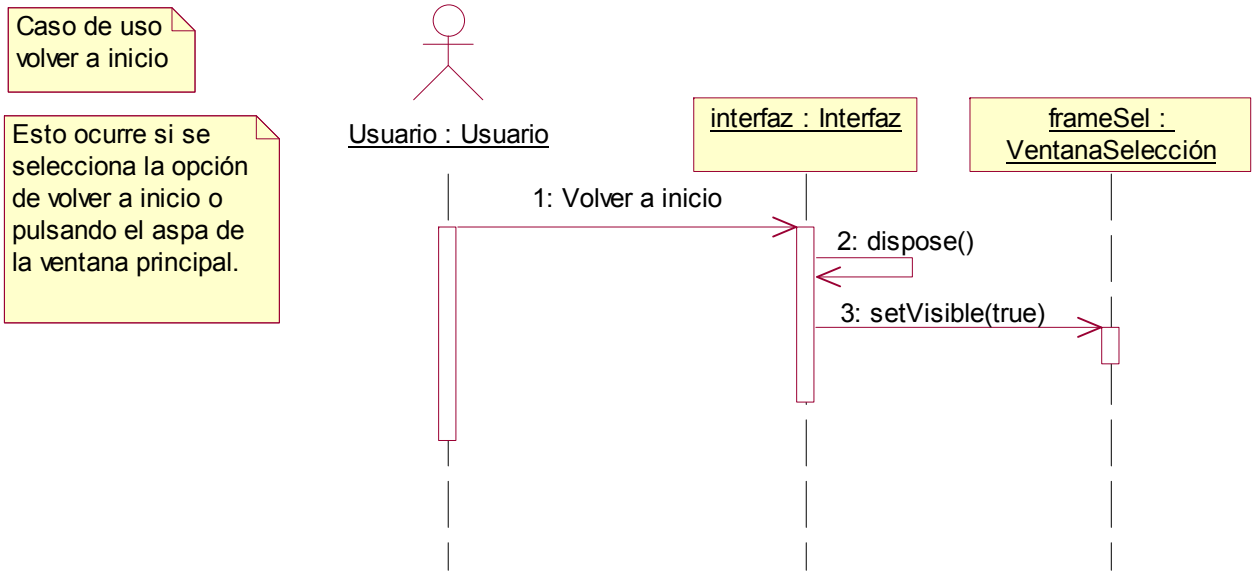
Un diagrama de secuencia muestra las interacciones entre objetos organizados en una secuencia temporal.

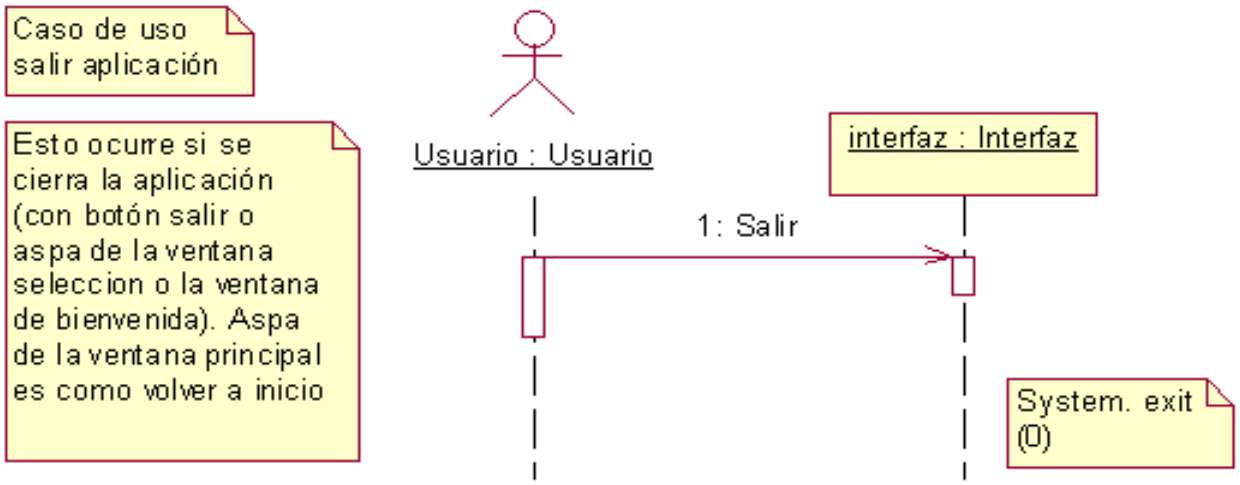
Como muestra, se presentan los siguientes diagramas de secuencia:

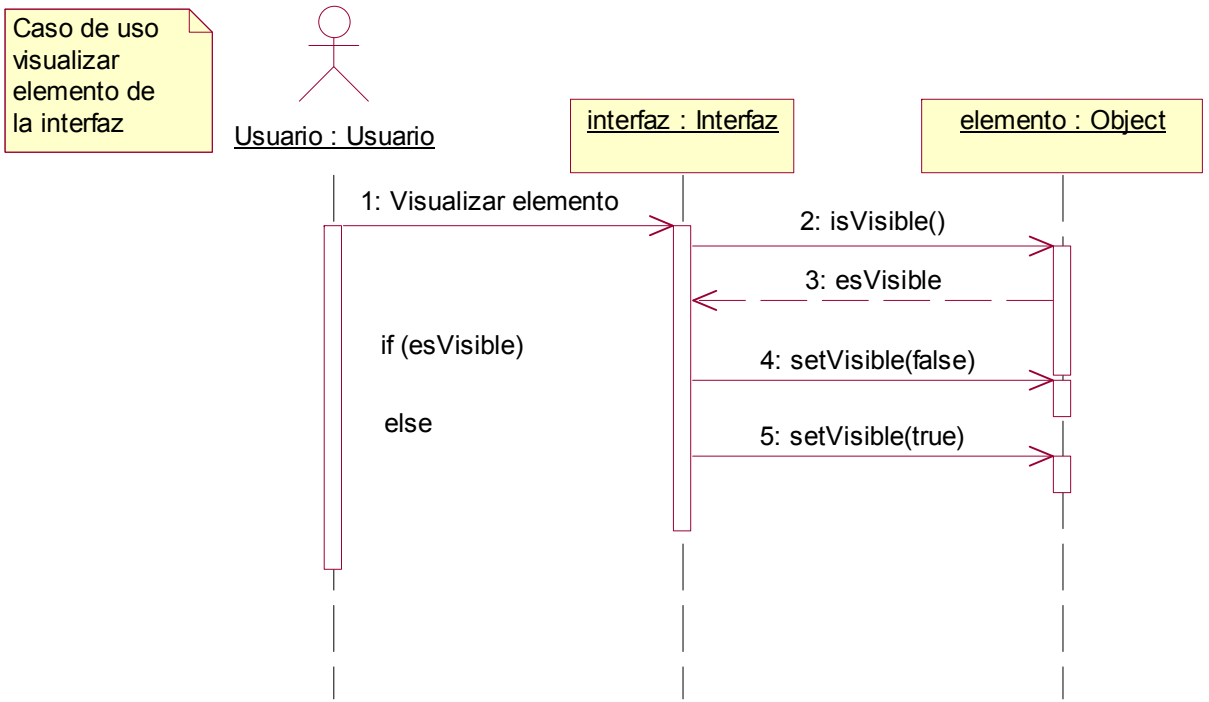
- Mostrar documento de bienvenida
- Iniciar aplicación
- Volver a Inicio
- Salir de la aplicación
- Visualizar elementos de la interfaz
- Crear pila
- Apilar
- Desapilar
- Consultar cima pila
- Consultar pila vacía
- Consultar tamaño pila
- Mostrar documentación estructura datos

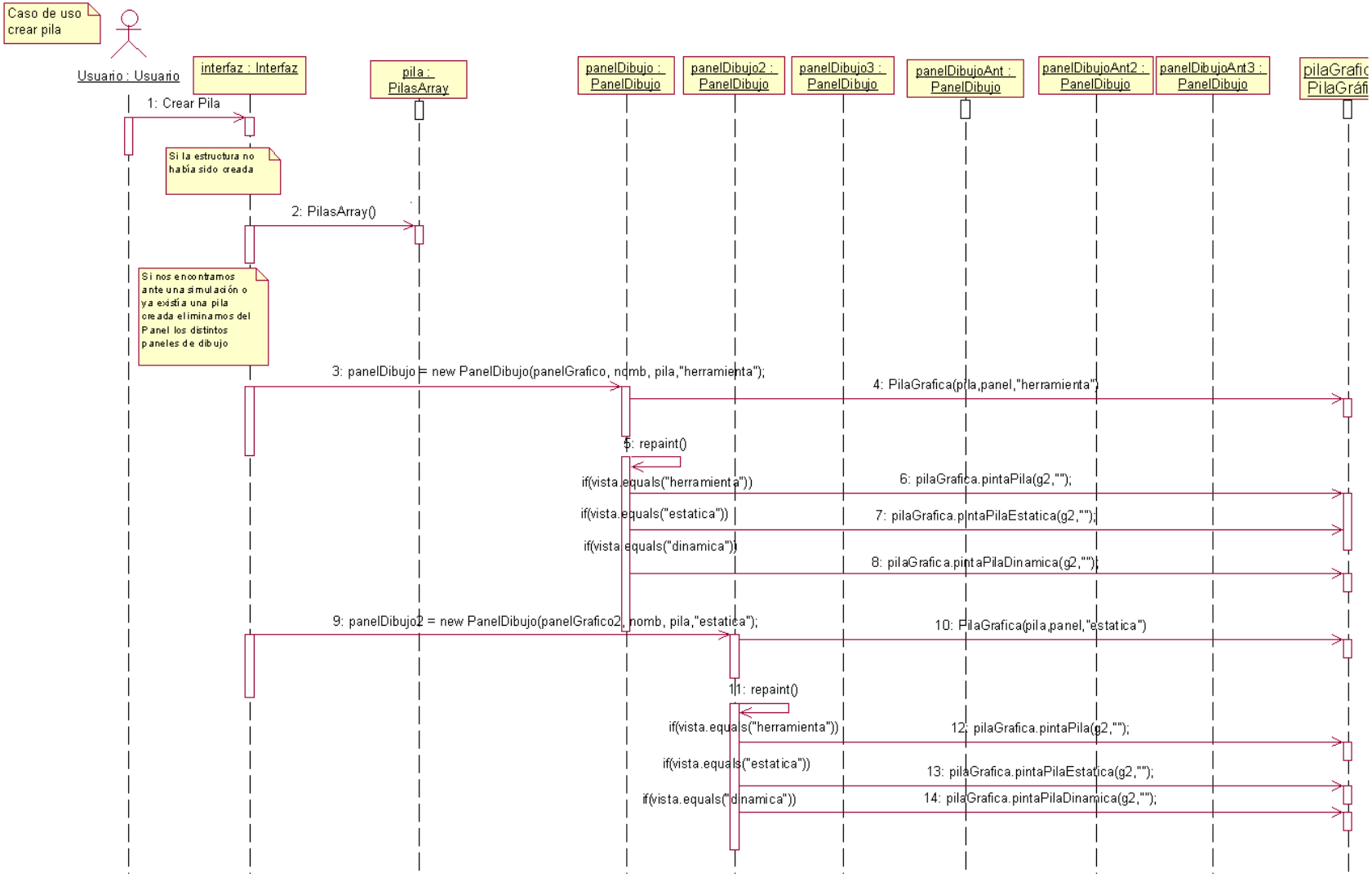

Caso de uso mostrar documento de bienvenida
Usuario : Usuario
ventanaInicio : VentanaInicio
cargador : CargadorDeHTMLs
ventanaSeleccion : VentanaSelección
1: ejecutar
2: CargadorDeHTMLs(this)
3: cargaPagina("Visualización y Animación de Estructuras de Datos y Métodos algorítmicos.htm",false);
4: Comenzar
5: ventanaSeleccion()

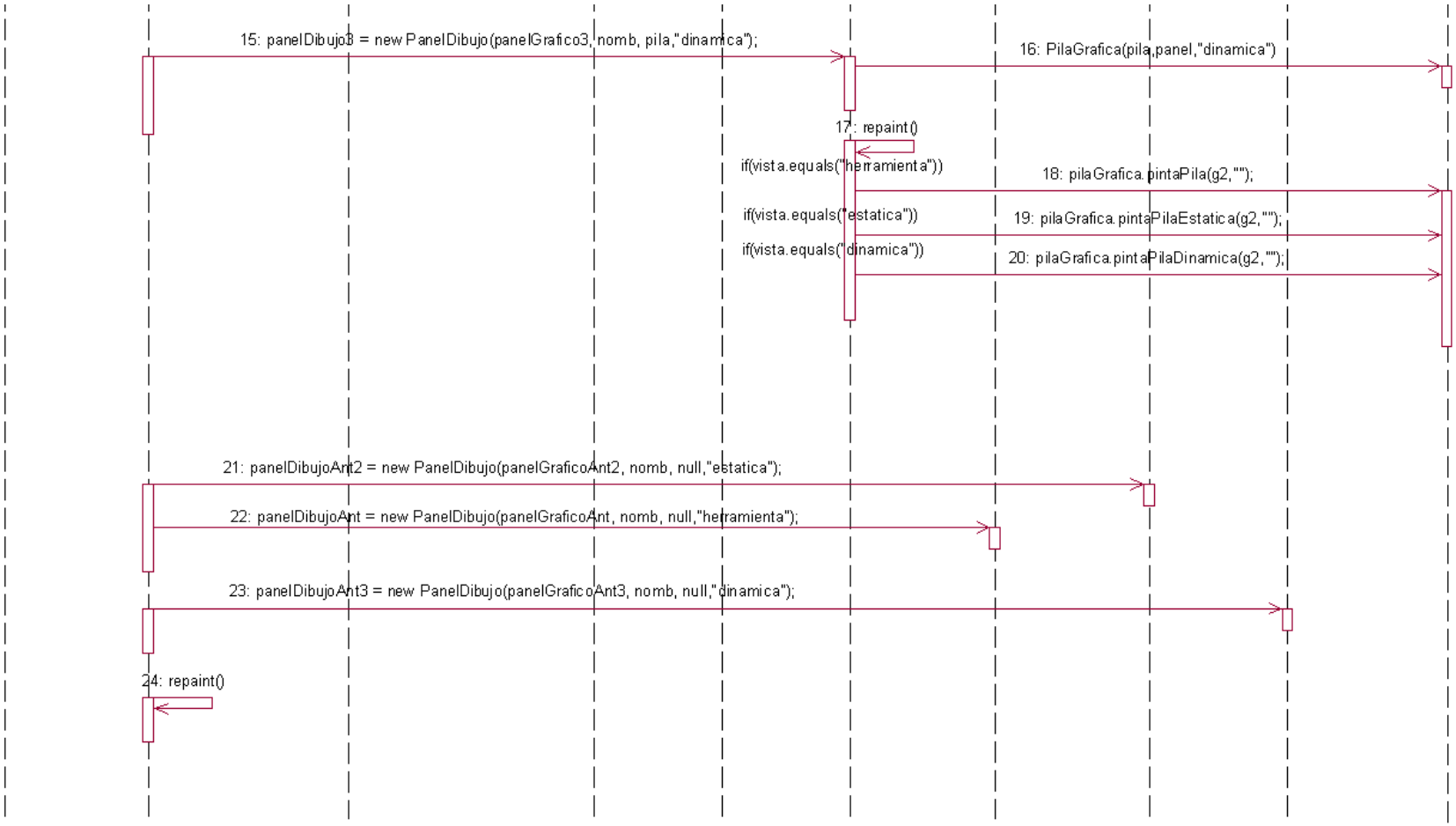

Caso de uso
iniciar
aplicación
Usuario : Usuario
ventanaSeleccion :
VentanaSelección
frame : Interfaz
1: Botón "Comenzar" en la pantalla de bienvenida (EDoALG, tipo)
Seleccionar si se desea visualizar
estructuras de datos o métodos
algorítmicos
Seleccionar el tipo de estructura o
método algorítmico que se desee
2: new Interfaz(tipo)

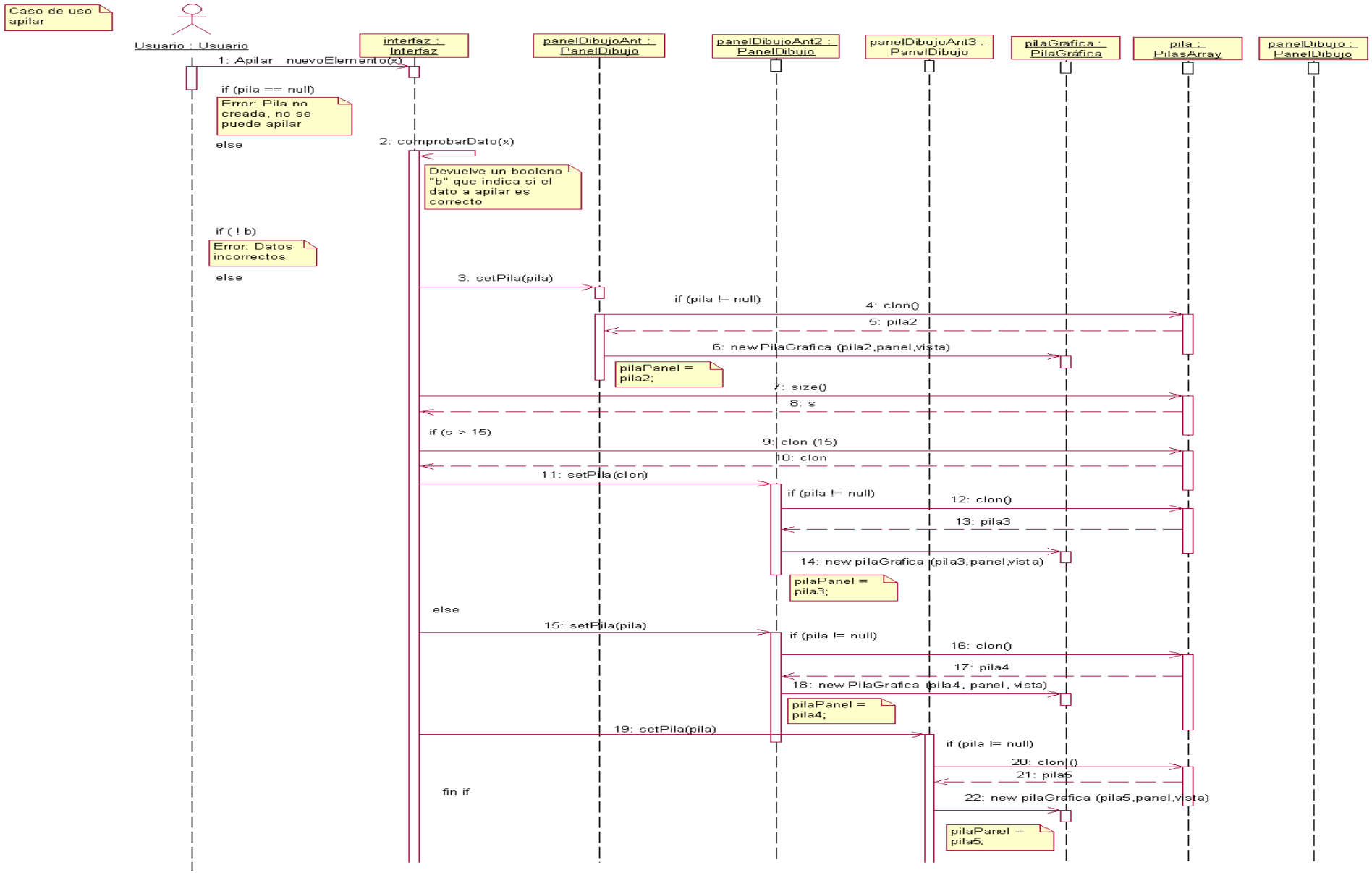

Caso de uso
volver a inicio
Esto ocurre si se selecciona la opción de volver a inicio o pulsando el aspa de la ventana principal.
Usuario : Usuario
interfaz : Interfaz
frameSel : VentanaSelección
1: Volver a inicio
2: dispose()
3: setVisible(true)

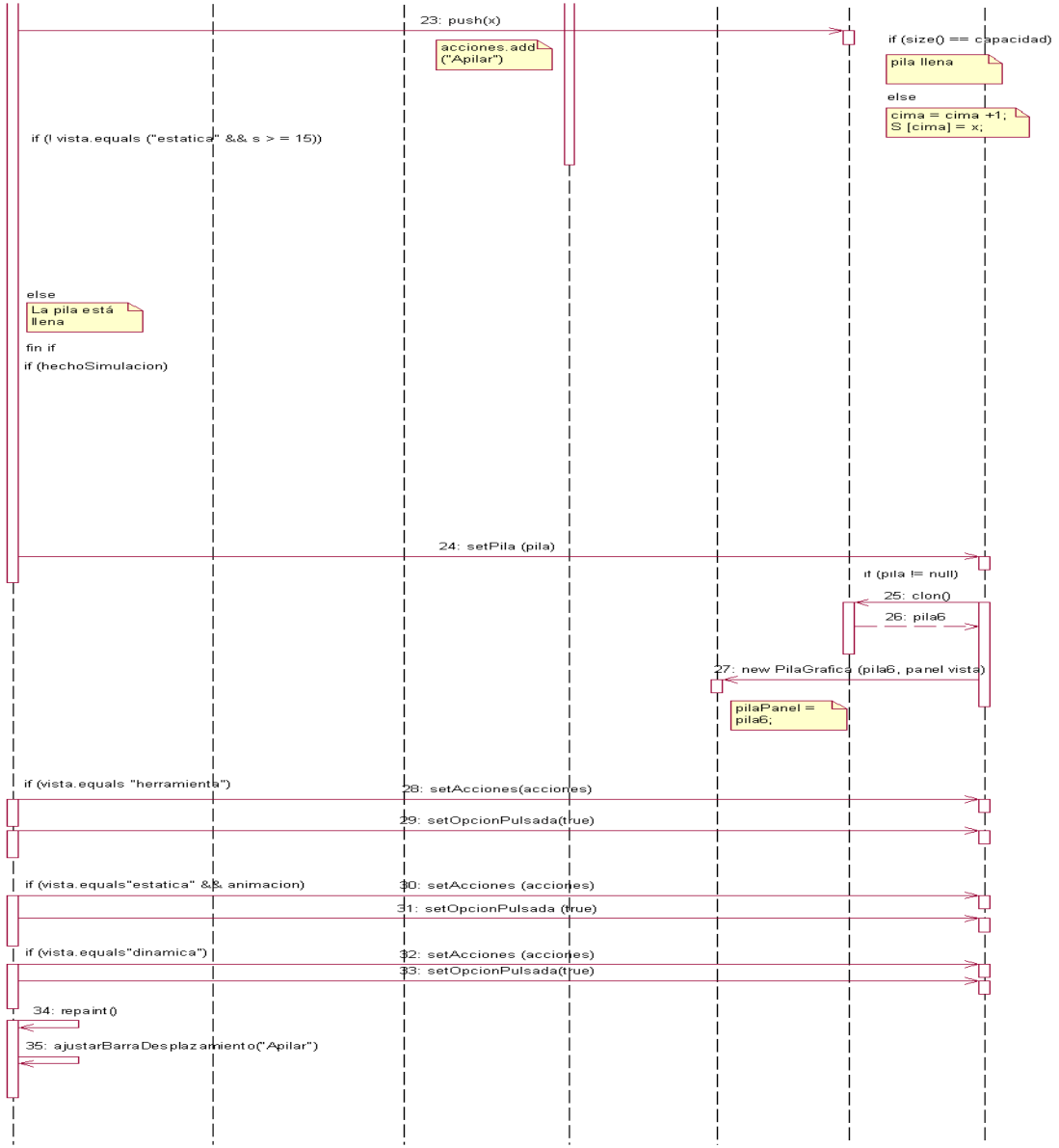

Caso de uso
salir aplicación
Esto ocurre si se cierra la aplicación (con botón salir o aspa de la ventana seleccion o la ventana de bienvenida). Aspa de la ventana principal es como volver a inicio
Usuario : Usuario
interfaz : Interfaz
1: Salir
System. exit (0)

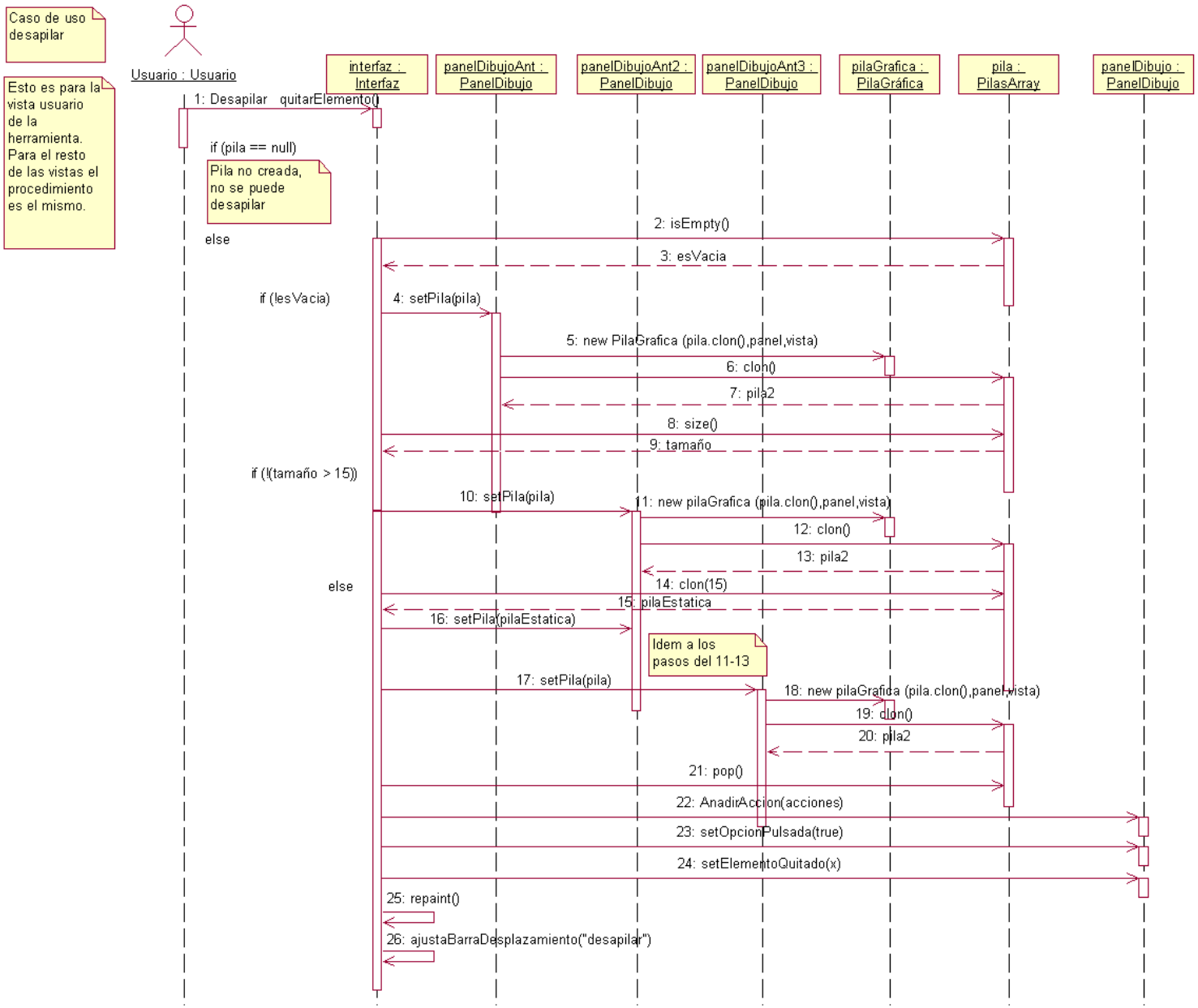

Caso de uso visualizar elemento de la interfaz
Usuario : Usuario
interfaz : Interfaz
elemento : Object
1: Visualizar elemento
2: isVisible()
3: esVisible
if (esVisible)
4: setVisible(false)
else
5: setVisible(true)

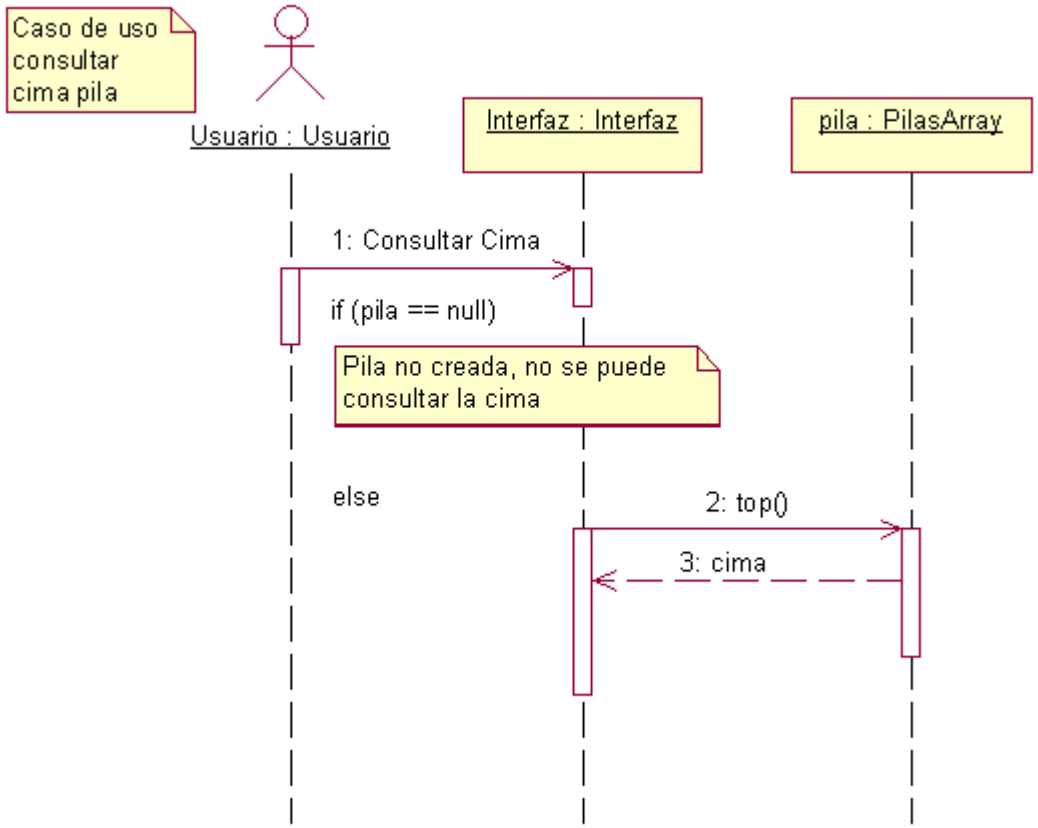

Caso de uso crear pila
Usuario : Usuario
interfaz : Interfaz
pila : PilasArray
panelDibujo : PanelDibujo
panelDibujo2 : PanelDibujo
panelDibujo3 : PanelDibujo
panelDibujoAnt : PanelDibujo
panelDibujoAnt2 : PanelDibujo
panelDibujoAnt3 : PanelDibujo
pilaGrafic : PilaGráfi
1: Crear Pila
Si la estructura no había sido creada
2: PilasArray()
Si nos encontramos ante una simulación o ya existía una pila creada eliminamos del Panel los distintos paneles de dibujo
3: panelDibujo = new PanelDibujo(panelGrafico, nomb, pila,"herramienta");
4: PilaGrafica(pila,panel,"herramienta")
5: repaint()
if(vista.equals("herramienta"))
6: pilaGrafica.pintaPila(g2,"");
if(vista.equals("estatica"))
7: pilaGrafica.pintaPilaEstatica(g2,"");
if(vista.equals("dinamica"))
8: pilaGrafica.pintaPilaDinamica(g2,"");
9: panelDibujo2 = new PanelDibujo(panelGrafico2, nomb, pila,"estatica");
10: PilaGrafica(pila,panel,"estatica")
11: repaint()
if(vista.equals("herramienta"))
12: pilaGrafica.pintaPila(g2,"");
if(vista.equals("estatica"))
13: pilaGrafica.pintaPilaEstatica(g2,"");
if(vista.equals("dinamica"))
14: pilaGrafica.pintaPilaDinamica(g2,"");

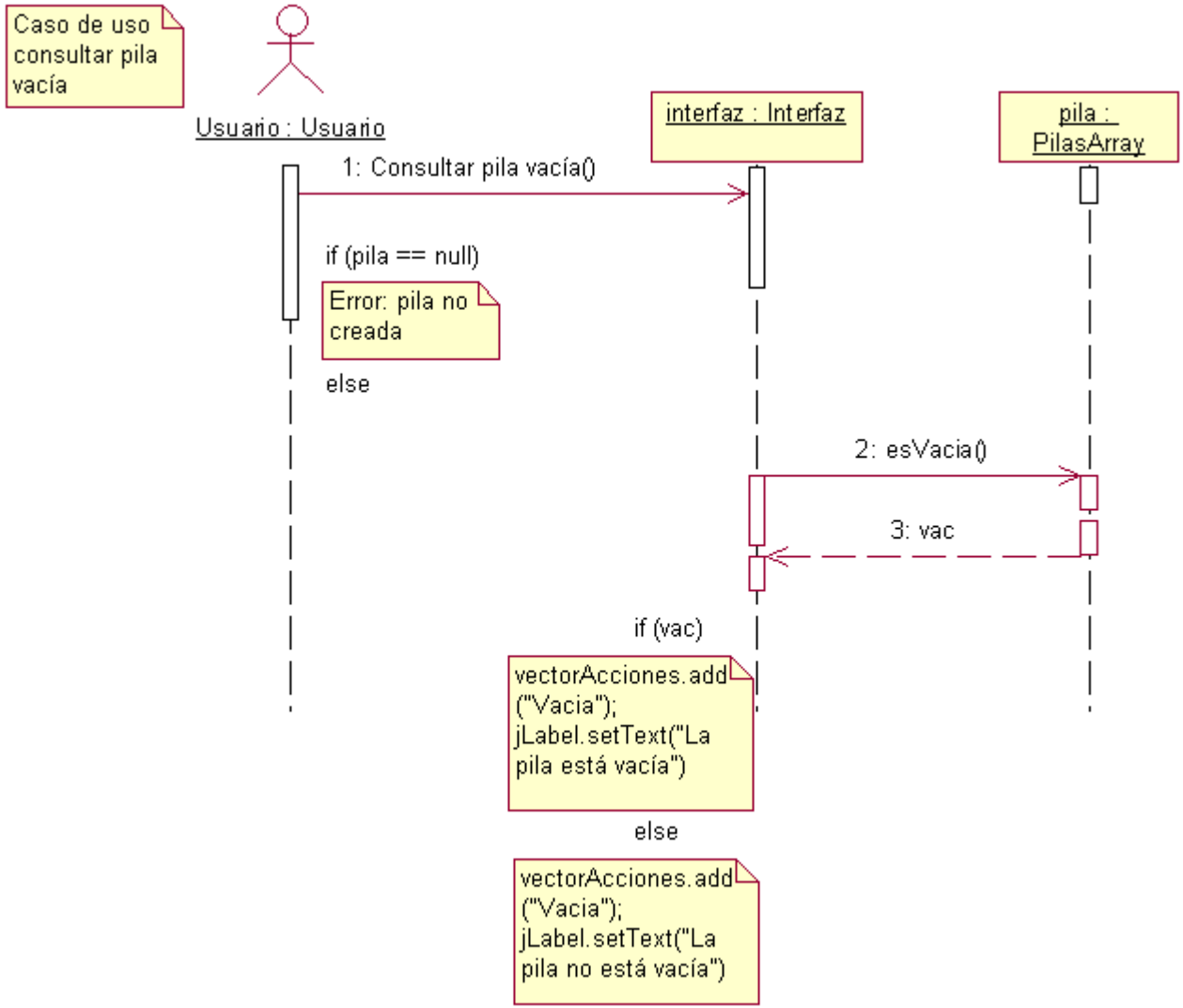

15: panelDibujo3 = new PanelDibujo(panelGrafico3,'nomb, pila,"dinamica");
16: PilaGrafica(pila,panel,"dinamica")
17: repaint()
if(vista.equals("herramienta"))
18: pilaGrafica.pintaPila(g2,"");
if(vista.equals("estatica"))
19: pilaGrafica.pintaPilaEstatica(g2,"");
if(vista.equals("dinamica"))
20: pilaGrafica.pintaPilaDinamica(g2,"");
21: panelDibujoAnt2 = new PanelDibujo(panelGraficoAnt2, nomb, null,"estatica");
22: panelDibujoAnt = new PanelDibujo(panelGraficoAnt, nomb, null,"herramienta");
23: panelDibujoAnt3 = new PanelDibujo(panelGraficoAnt3, nomb, null,"dinamica");
24: repaint()

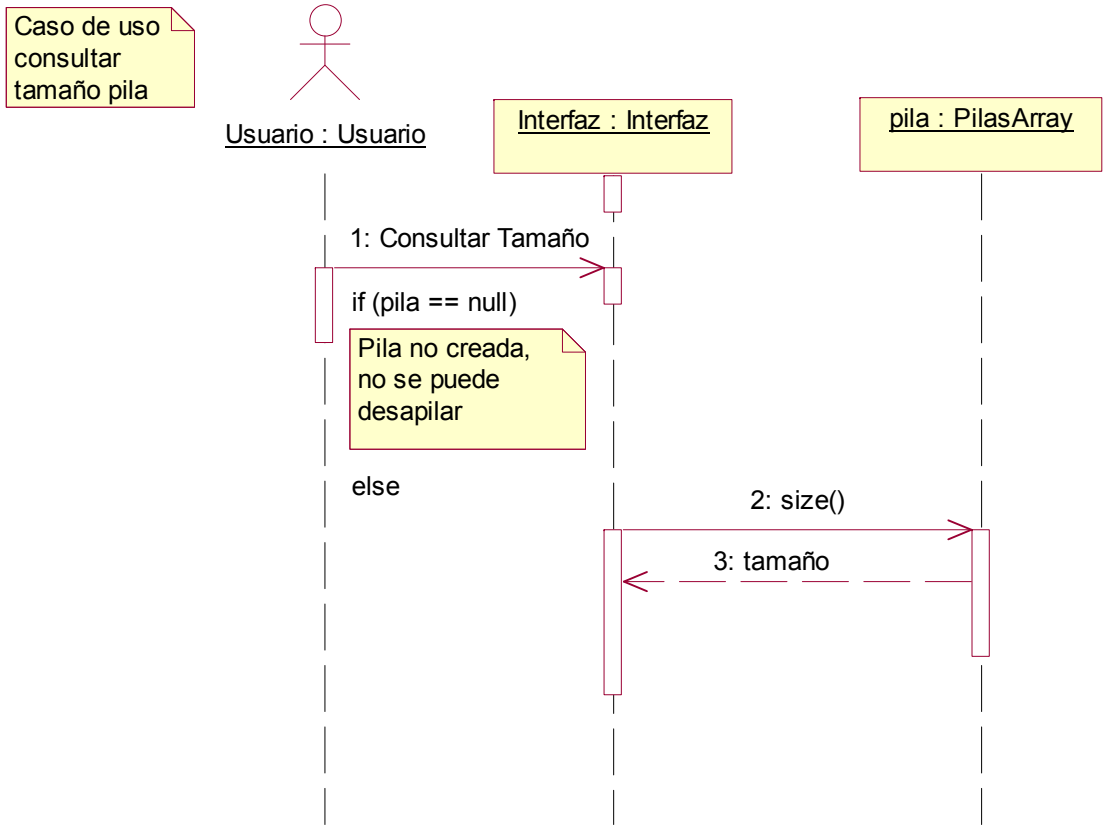

Caso de uso apilar
Usuario : Usuario
interfaz : Interfaz
panelDibujoAnt : PanelDibujo
panelDibujoAnt2 : PanelDibujo
panelDibujoAnt3 : PanelDibujo
pilaGrafica : PilaGráfica
pila : PilasArray
panelDibujo : PanelDibujo
1: Apilar nuevoElemento(x)
if (pila == null)
Error: Pila no creada, no se puede apilar
else
2: comprobarDato(x)
Devuelve un booleno "b" que indica si el dato a apilar es correcto
if (! b)
Error: Datos incorrectos
else
3: setPila(pila)
if (pila != null)
4: clon()
5: pila2
6: newPilaGrafica (pila2,panel,vista)
pilaPanel = pila2;
7: size()
8: s
if (s > 15)
9: clon (15)
10: clon
11: setPila(clon)
if (pila != null)
12: clon()
13: pila3
14: new pilaGrafica (pila3,panel,vista)
pilaPanel = pila3;
else
15: setPila(pila)
if (pila != null)
16: clon()
17: pila4
18: new PilaGrafica (pila4, panel, vista)
pilaPanel = pila4;
19: setPila(pila)
if (pila != null)
20: clon()
21: pila5
fin if
22: new pilaGrafica (pila5,panel,vista)
pilaPanel = pila5;

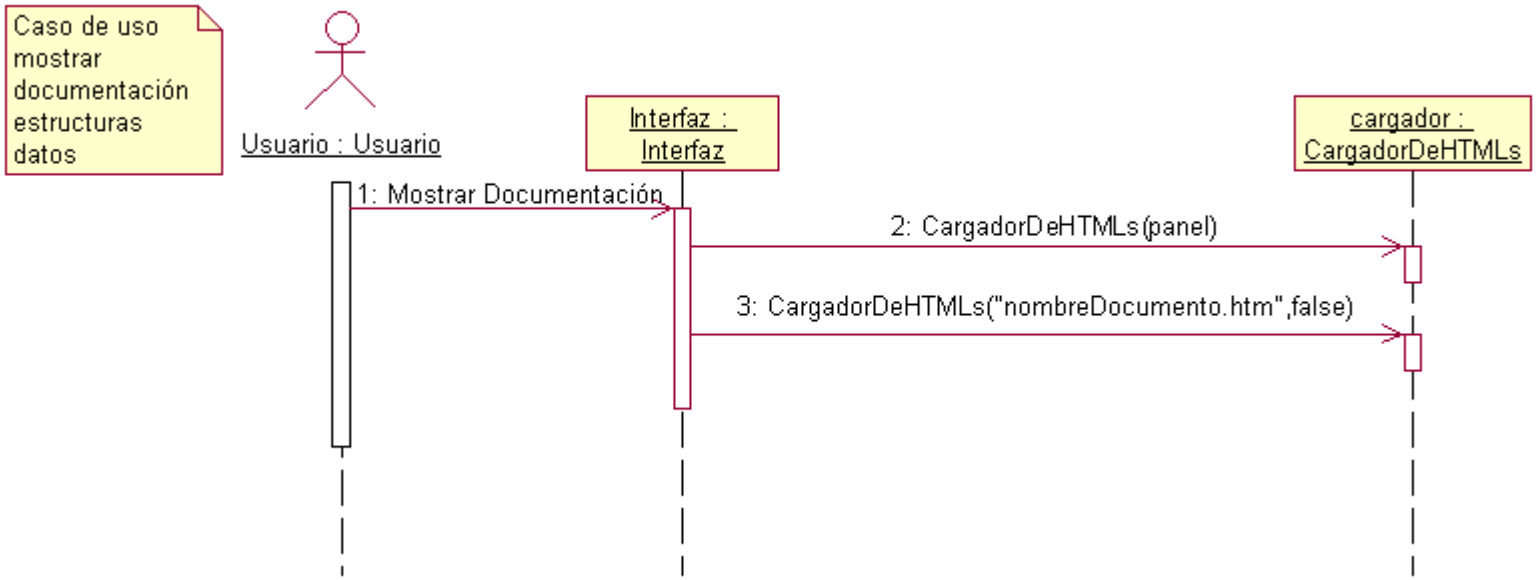

23: push(x)
acciones.add ("Apilar")
if (size() == capacidad)
pila llena
else
cima = cima +1;
S [cima] = x;
if (! vista.equals ("estatica" && s > = 15))
else
La pila está llena
fin if
if (hechoSimulacion)
24: setPila (pila)
if (pila != null)
25: clon()
26: pila6
27: new PilaGrafica (pila6, panel vista)
pilaPanel = pila6;
if (vista.equals "herramienta")
28: setAcciones(acciones)
29: setOpcionPulsada(true)
if (vista.equals"estatica" && animacion)
30: setAcciones (acciones)
31: setOpcionPulsada (true)
if (vista.equals"dinamica")
32: setAcciones (acciones)
33: setOpcionPulsada(true)
34: repaint()
35: ajustarBarraDesplazamiento("Apilar")

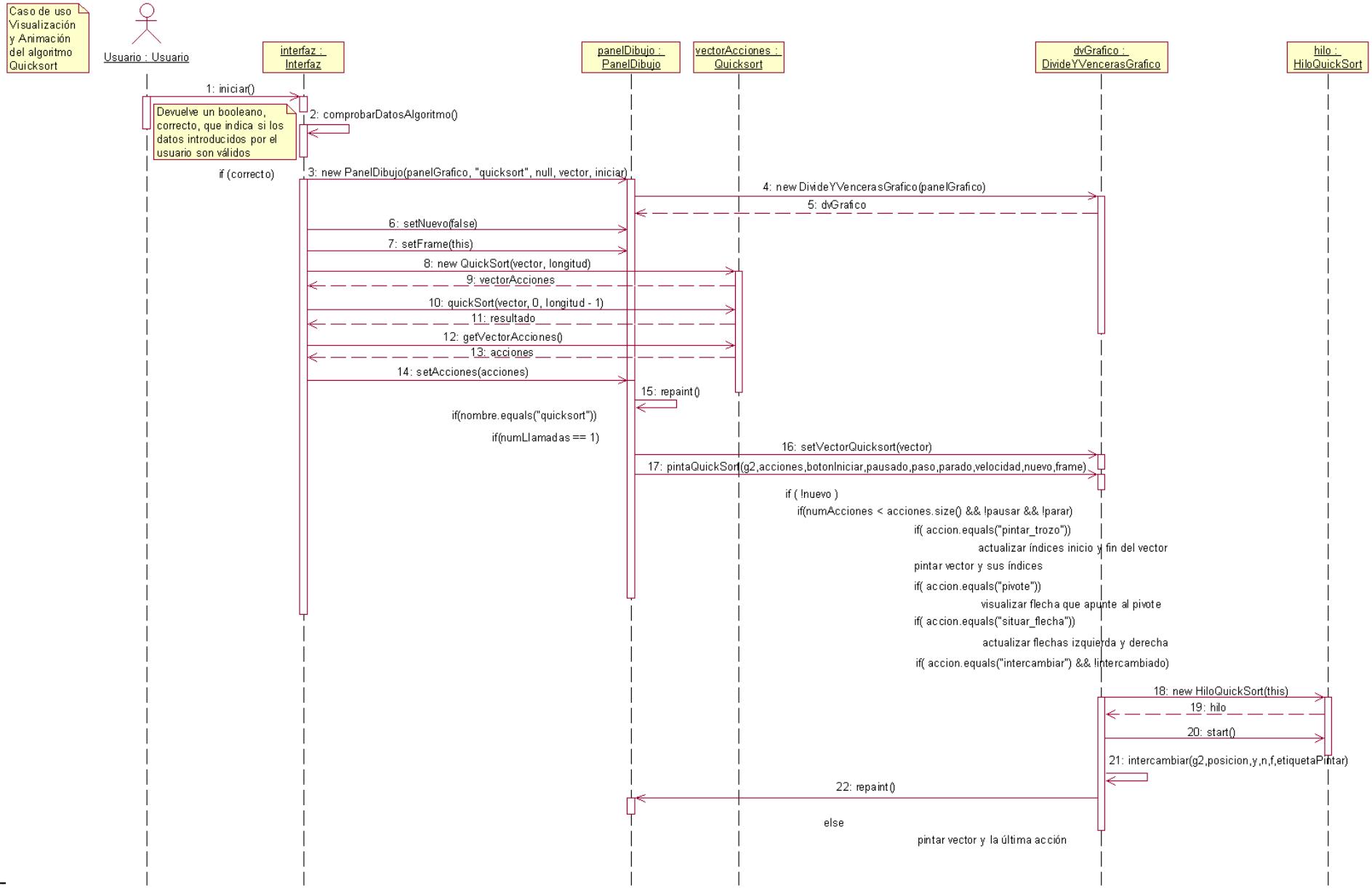

Caso de uso desapilar
Esto es para la vista usuario de la herramienta. Para el resto de las vistas el procedimiento es el mismo.
Usuario : Usuario
interfaz : Interfaz
panelDibujoAnt : PanelDibujo
panelDibujoAnt2 : PanelDibujo
panelDibujoAnt3 : PanelDibujo
pilaGrafica : PilaGráfica
pila : PilasArray
panelDibujo : PanelDibujo
1: Desapilar quitarElemento()
if (pila == null)
Pila no creada, no se puede desapilar
else
2: isEmpty()
3: esVacia
if (!esVacia)
4: setPila(pila)
5: new PilaGrafica (pila.clon(),panel,vista)
6: clon()
7: pila2
8: size()
9: tamaño
if (!(tamaño > 15))
10: setPila(pila)
11: new pilaGrafica (pila.clon(),panel,vista)
12: clon()
13: pila2
else
14: clon(15)
15: pilaEstatica
16: setPila(pilaEstatica)
Idem a los pasos del 11-13
17: setPila(pila)
18: new pilaGrafica (pila.clon(),panel,vista)
19: clon()
20: pila2
21: pop()
22: AnadirAccion(acciones)
23: setOpcionPulsada(true)
24: setElementoQuitado(x)
25: repaint()
26: ajustaBarraDesplazamiento("desapilar")


Caso de uso
consultar
cima pila
Usuario : Usuario
Interfaz : Interfaz
pila : PilasArray
1: Consultar Cima
if (pila == null)
Pila no creada, no se puede
consultar la cima
else
2: top()
3: cima


Caso de uso consultar pila vacía
Usuario : Usuario
interfaz : Interfaz
pila : PilasArray
1: Consultar pila vacía()
if (pila == null)
Error: pila no creada
else
2: esVacia()
3: vac
if (vac)
vectorAcciones.add ("Vacia");
jLabel.setText("La pila está vacía")
else
vectorAcciones.add ("Vacia");
jLabel.setText("La pila no está vacía")


Caso de uso
consultar
tamaño pila
Usuario : Usuario
Interfaz : Interfaz
pila : PilasArray
1: Consultar Tamaño
if (pila == null)
Pila no creada,
no se puede
desapilar
else
2: size()
3: tamaño

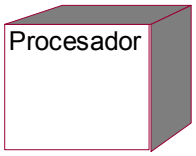

Caso de uso
mostrar
documentación
estructuras
datos
Usuario : Usuario
Interfaz :
Interfaz
cargador :
CargadorDeHTMLs
1: Mostrar Documentación
2: CargadorDeHTMLs(panel)
3: CargadorDeHTMLs("nombreDocumento.htm",false)


Caso de uso Visualización y Animación del algoritmo Quicksort
Usuario : Usuario
interfaz : Interfaz
panelDibujo : PanelDibujo
vectorAcciones : Quicksort
dvGrafico : DivideYVencerasGrafico
hilo : HiloQuickSort
1: iniciar()
Devuelve un booleano, correcto, que indica si los datos introducidos por el usuario son válidos
2: comprobarDatosAlgoritmo()
if (correcto)
3: new PanelDibujo(panelGrafico, "quicksort", null, vector, iniciar)
4: new DivideYVencerasGrafico(panelGrafico)
5: dvGrafico
6: setNuevo(false)
7: setFrame(this)
8: new QuickSort(vector, longitud)
9: vectorAcciones
10: quickSort(vector, 0, longitud - 1)
11: resultado
12: getVectorAcciones()
13: acciones
14: setAcciones(acciones)
15: repaint()
if(nombre.equals("quicksort"))
if(numLlamadas == 1)
16: setVectorQuicksort(vector)
17: pintaQuickSort(g2,acciones,botonIniciar,pausado,paso,parado,velocidad,nuevo,frame)
if (!nuevo)
if(numAcciones < acciones.size() && !pausar && !parar)
if(accion.equals("pintar_trozo"))
actualizar índices inicio y fin del vector
pintar vector y sus índices
if(accion.equals("pivote"))
visualizar flecha que apunte al pivote
if(accion.equals("situar_flecha"))
actualizar flechas izquierda y derecha
if(accion.equals("intercambiar") && !intercambiado)
18: new HiloQuickSort(this)
19: hilo
20: start()
21: intercambiar(g2,posicion,y,n,f,etiquetaPintar)
22: repaint()
else
pintar vector y la última acción

6. Diagrama de Despliegue

Un diagrama de despliegue muestra las relaciones físicas entre los componentes de software y de hardware del sistema, visualizando la distribución de los distintos componentes.

En nuestro caso, el diagrama de despliegue resulta ser trivial, ya que el único elemento a representar es el procesador.



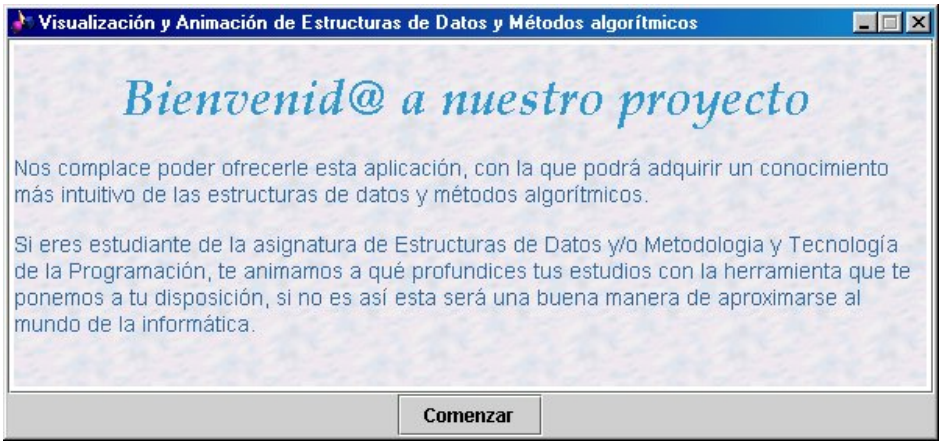
1. Introducción

En el campo de la informática las estructuras de datos y los algoritmos son dos áreas fundamentales. Las estructuras de datos nos ayudan a estructurar la información que manejamos mientras que los métodos algoritmicos a resolver más fácilmente problemas. Estas dos áreas se enseñan en las asignaturas de Estructuras de Datos y/o Metodología y Tecnología de la Programación respectivamente. Actualmente los alumnos y profesores de las facultades de informática usan como sistema de aprendizaje y enseñanza, respectivamente, documentación escrita. El modelo lógico de enseñanza actual muchas veces resulta poco ilustrativo al alumno puesto que es estático. Por todo ello nos complace poder ofrecerle esta aplicación que podrá usarse una guía visual animada complementaria a la documentación escrita , con la que podrá adquirir un conocimiento más intuitivo de las estructuras de datos y métodos algorítmicos

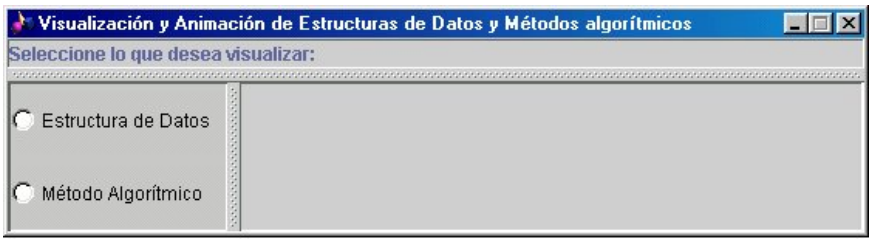
Si es estudiante de la asignatura de Estructuras de Datos y/o Metodología y Tecnología de la Programación, le animamos a qué profundice sus estudios con la herramienta que te ponemos a tu disposición, si no es así esta será una buena manera de aproximarse al mundo de la informática.

A continuación se proporciona una ayuda sobre el uso de la herramienta para ayudar al usuario a sacar el máximo partido de la misma.

La primera pantalla que aparecerá al abrir la aplicación es una pantalla introductoria en la que se explica la objetivo de la herramienta y los destinatarios a los que va dirigida.

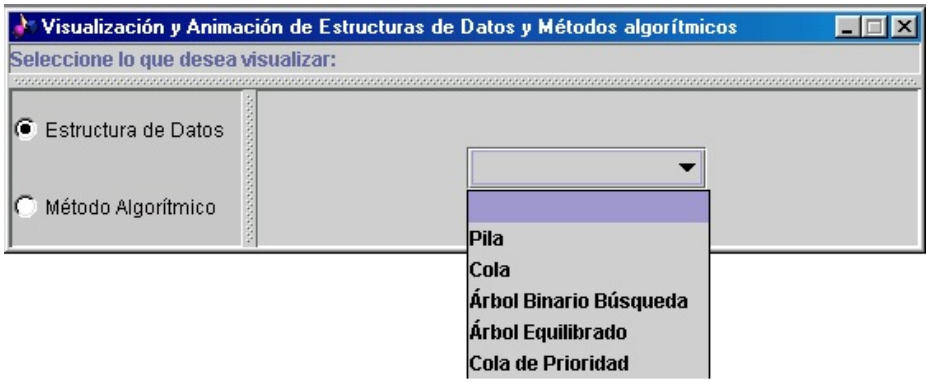


Pulse aceptar y se mostrará una ventana en la que el usuario puede seleccionar si se desea trabajar con una estructura de datos o con un algoritmo.



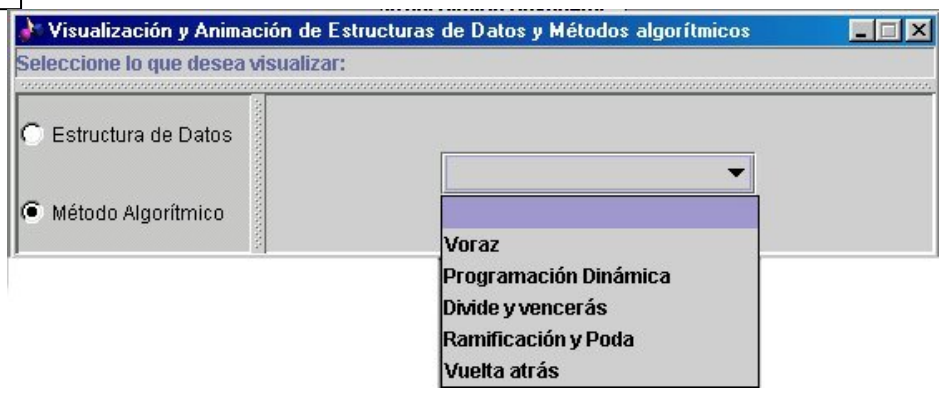
Si selecciona la opción ED y aparecerá un desplegable en el que se mostrarán las EDs disponibles:

- Pila
- Cola
- Árbol binario de búsqueda
- Árbol equilibrado AVL
- Cola de prioridad de mínimos



Si selecciona la opción algoritmo aparecerá un desplegable en el que se mostrarán los métodos algorítmicos disponibles:

- Voraz
- Programación dinámica
- Divide y vencerás
- Ramificación y poda
- Vuelta atrás



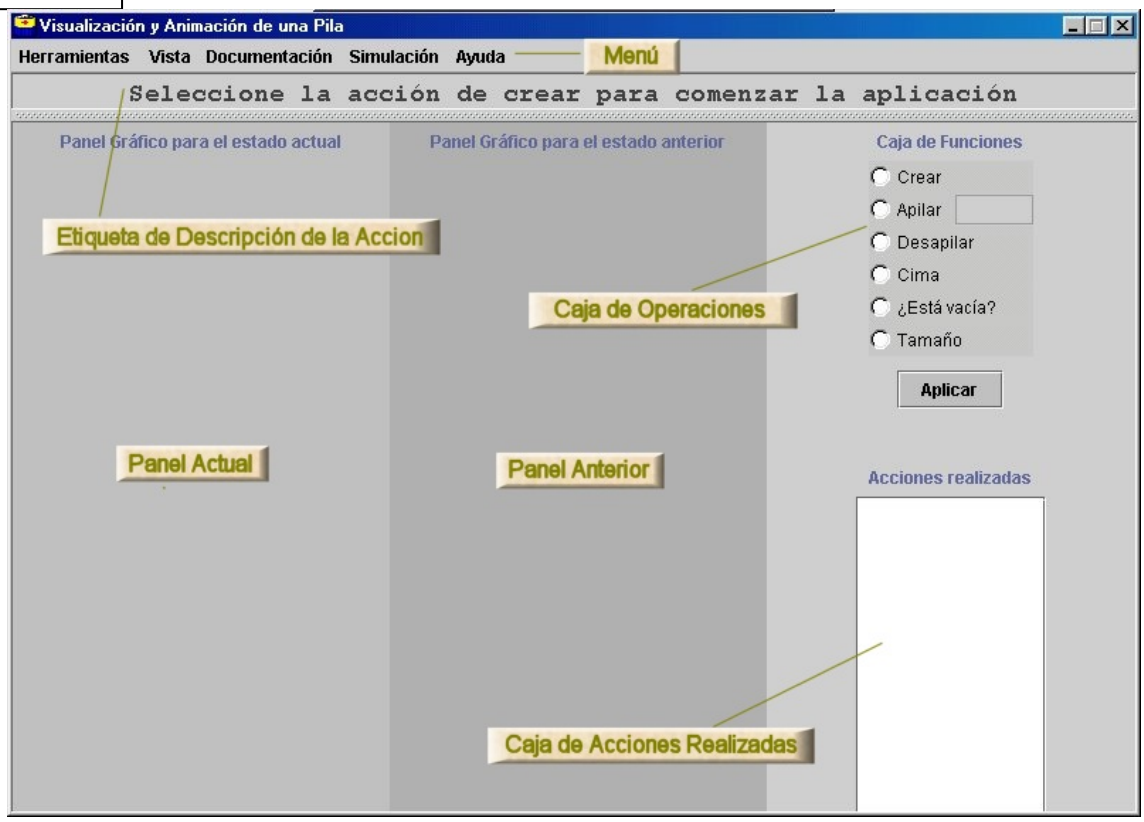
Seleccione el método algorítmico o la estructura de datos con la que desee trabajar.

2. Ejecución de la Aplicación

2.1. Estructuras de datos

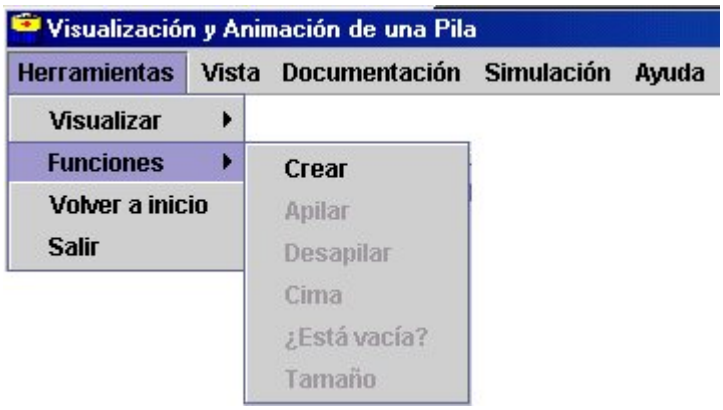
En la ventana de trabajo de una estructura de datos podemos distinguir las siguientes partes:

- Panel de vista actual: panel en el que se visualiza el estado actual de la ED, es decir, se visualiza la ED tras la realización de la última operación. En este panel también observaremos las animaciones.
- Panel de vista anterior, se muestra la estructura de datos en el estado anterior al actual.
- Caja de funciones. Consta de dos partes diferenciadas:
  - **Caja de operaciones**: Caja dónde se puede seleccionar las operaciones a realizar en la ED. Si se selecciona sobre una función no aplicable en ese momento se mostrará un mensaje de error. Tras seleccionar la operación a realizar se ha de pulsar el botón aplicar para indicar la aceptación de la opción seleccionada.
  - **Caja de acciones realizadas**: Muestra las acciones realizadas sobre la estructura de datos hasta el momento.
- Etiqueta de descripción de la acción, en la que se indica la operación que se está realizando en el momento actual.



- Menú
  - Herramientas
    - Visualizar: activa o desactiva la visualización de las distintas partes de la pantalla: etiqueta descriptiva, panel actual, anterior...

    - Volver a Inicio: vuelve a la pantalla de selección
    - Salir
    - Funciones aplicables. A diferencia con la caja de funciones, en este menú sólo aparecerán activadas las funciones aplicables.



- Vistas. Hay dos tipos:
  - Vista usuario de la herramienta: muestra el comportamiento de la ED independientemente de la implementación
  - Vista usuario de desarrollo. Representa el comportamiento de la ED con relación a la implementación.

    Encontraremos las siguientes vistas:
    - Vista Estática: se corresponde a la implementación estática de la estructura.
    - Vista Dinámica: Se corresponde a la implementación dinámica de la estructura.

- Documentación de la ED
  - Código: Si nos hayamos en la vista estática aparecerá la implementación estática, si estamos en la vista dinámica se mostrará el código de la implementación dinámica y si estamos en la vista de usuario de la herramienta se mostrarán ambas implementaciones.
  - Coste: coste comparativo de la ejecución de las distintas operaciones en las distintas implementaciones.
  - Especificación: Especificación algebraica de la estructura.
  - Ayuda adicional acerca de la ED y sus operaciones y usos.

- Simulación: realiza la simulación de la ejecución de un conjunto de operaciones sobre la ED. La opción simulación se realiza sobre el panel que representa la vista actual de la estructuras. Esta opción se encuentra en todas las estructuras en la vista de usuario de la herramienta, salvo en el montículo que se encuentra en la implementación de usuario de desarrollo en la implementación dinámica.

- Ayuda:
  - Índice de materias: Contiene un catálogo de información que contiene todos los temas de ayuda disponibles, para facilitar el entendimiento y posibilitar la consulta, de cualquiera de las estructuras de datos y esquemas algorítmicos utilizados en la aplicación.
  - Manual de usuario: Ayuda acerca del uso de la herramienta

2.1.1. Estructura de datos Pila

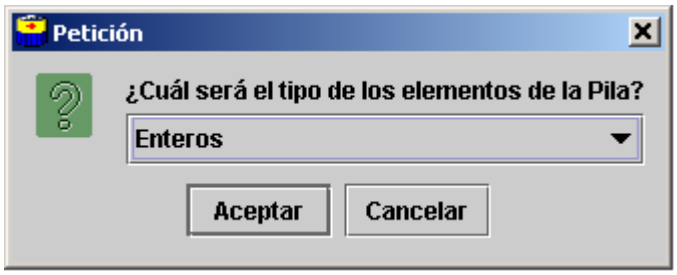
Las operaciones disponibles en la estructura de datos pila son:

- Crear
- Apilar un elemento
- Desapilar un elemento
- Consultar la cima de la pila
- Consultar el tamaño de la pila
- Consultar si la pila está vacía

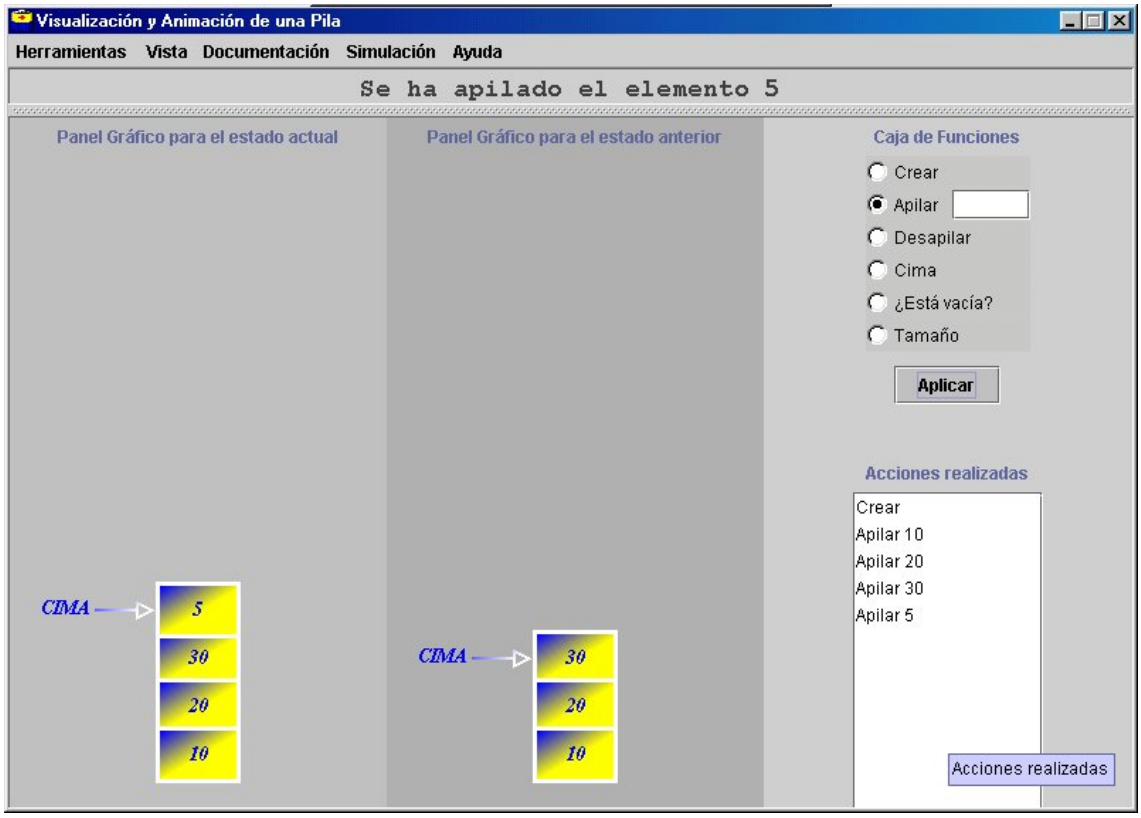
Para aplicar estas operaciones el usuario dispone de dos posibilidades:

- Seleccionar la operación desde el menú herramientas → funciones
- Seleccionar la operación desde la caja de operaciones. Si se opta por esta opción una vez seleccionada la operación a aplicar se deberá pulsar el botón aplicar.

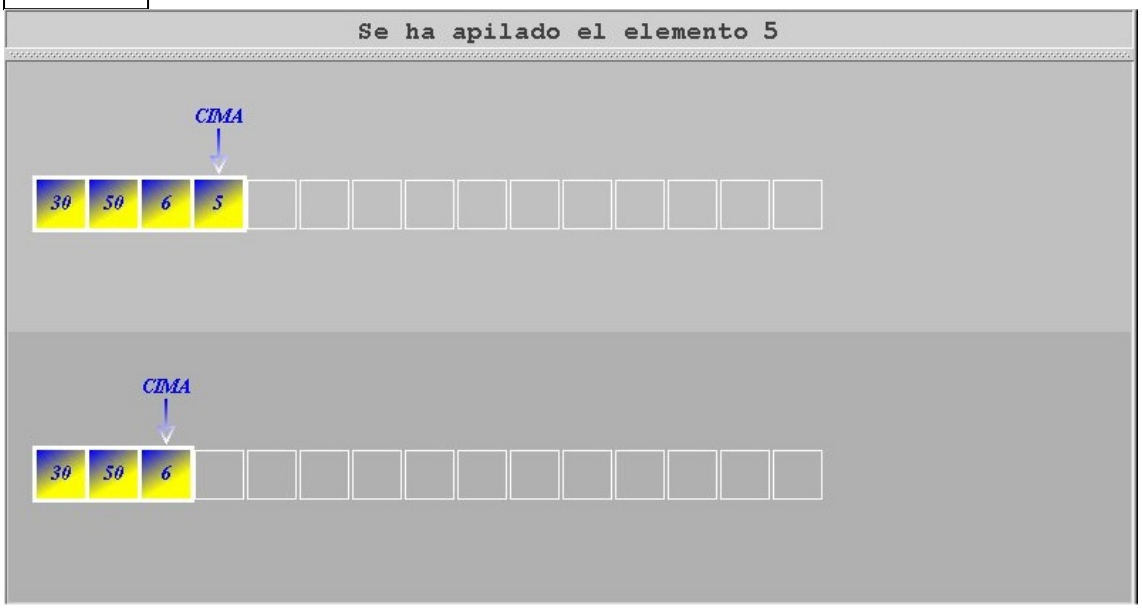
Al crear la estructura de datos, se mostrará una ventana que solicite el tipo de datos de los elementos que compondrán la estructura.



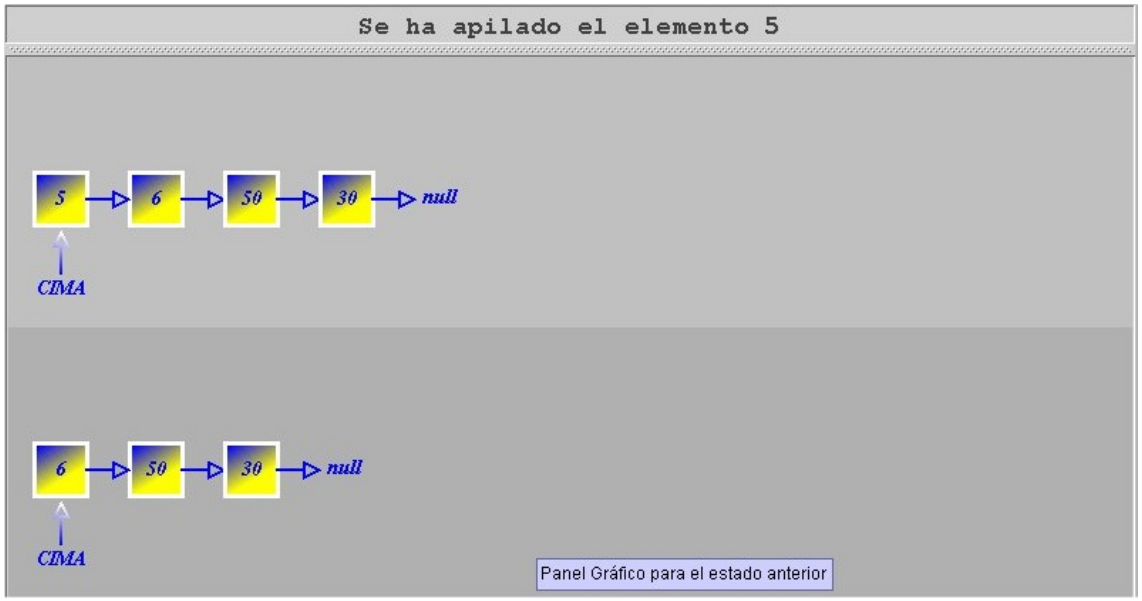
La representación de la pila en la vista usuario de la herramienta la de una pila general, formada por nodos que crecen o decrecen verticalmente. La cima de la pila se señalizará con una flecha.



Si el usuario selecciona en la vista usuario de desarrollo la implementación estática, la representación que podrá visualizar será la de una pila implementada mediante un vector.



Si el usuario selecciona en la vista usuario de desarrollo la implementación dinámica, la representación que podrá visualizar será la una estructura pila implementada mediante lista enlazada.



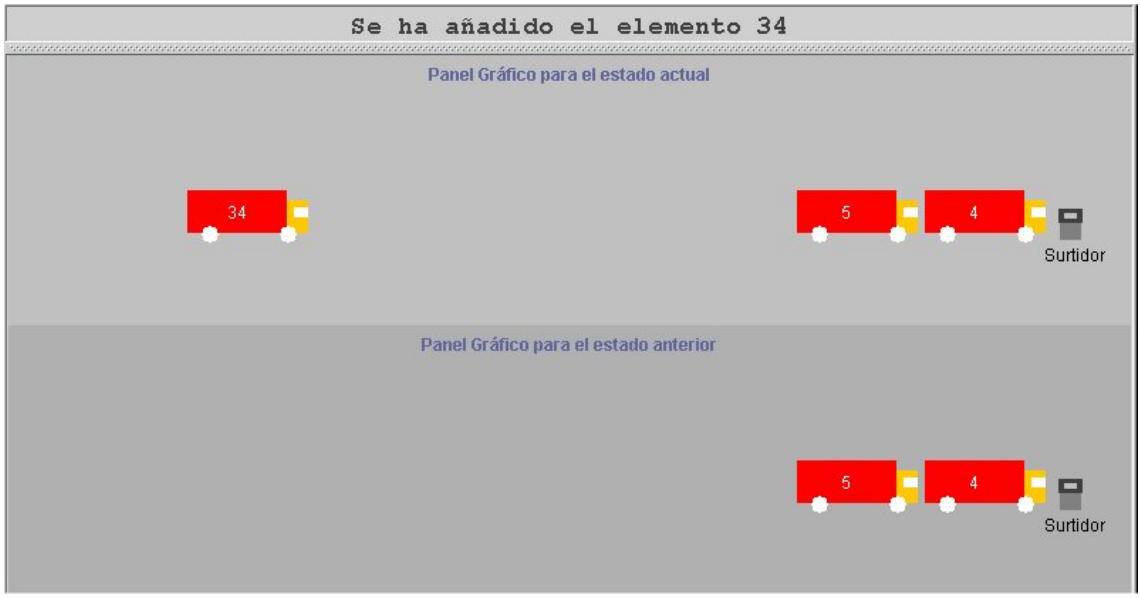
2.1.2. Estructura de datos Cola

Las operaciones disponibles en la estructura de datos pila son:

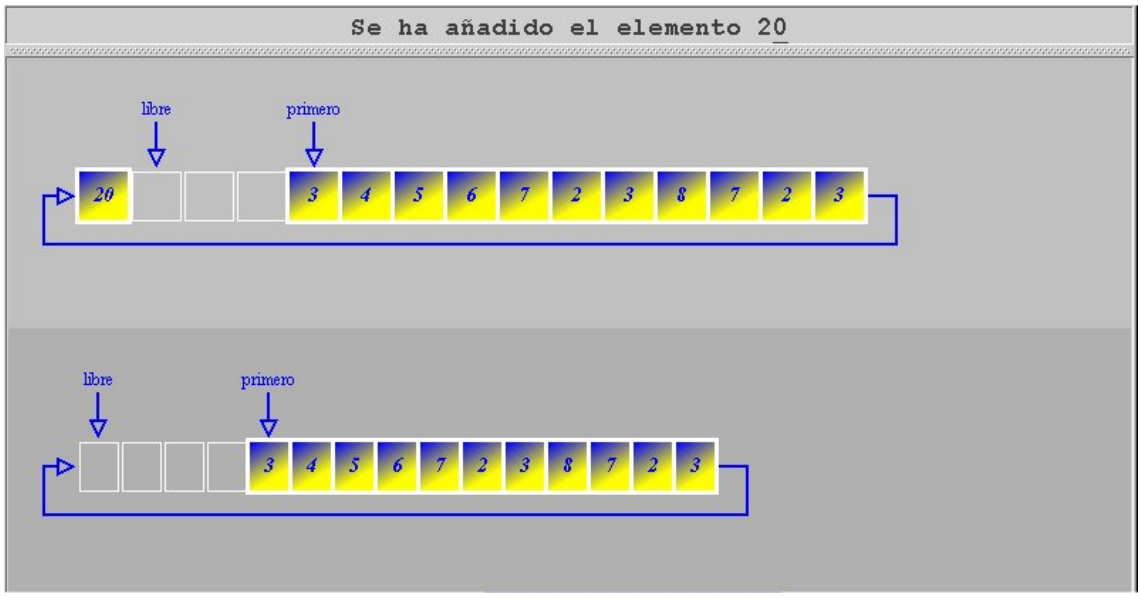
- Crear
- Añadir
- Eliminar
- Consultar el primero
- Consultar si la cola está vacía

- Consultar el tamaño de la cola

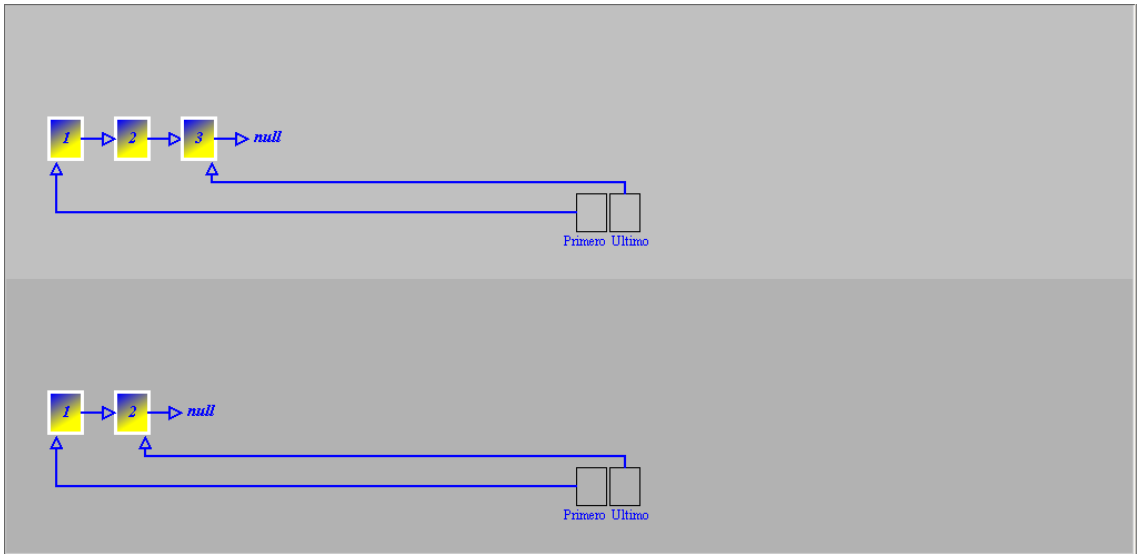
La representación de la cola en la vista usuario de la herramienta será una cola de camiones que esperan repostar en una gasolinera.



Si el usuario selecciona en la vista usuario de desarrollo la implementación estática, la representación que podrá visualizar será la implementación de la cola como un vector circular con dos índices, primero y libre, que apuntarán a la posición del primer elemento de la cola y a la primera posición libre, respectivamente.



Si el usuario selecciona en la vista usuario de desarrollo la implementación dinámica, la representación que podrá visualizar de la estructura cola será la implementación de la misma mediante una lista enlazada:



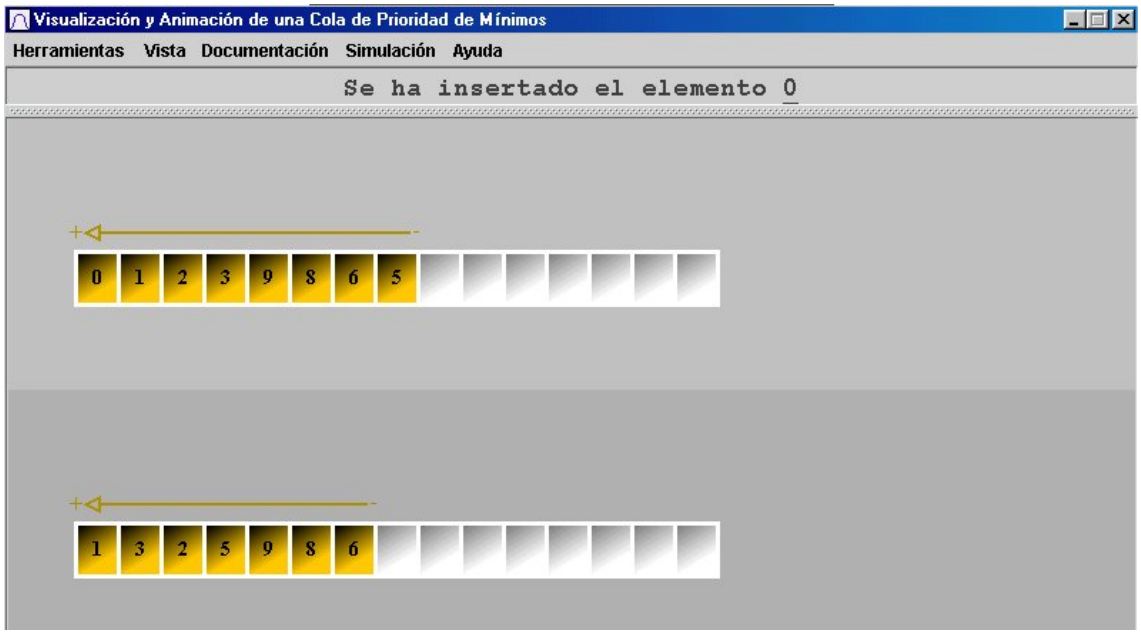
2.1.3. Estructura de datos Cola de Prioridad

La cola de prioridad de mínimos dispone de las siguientes operaciones:

- Crear
- Insertar
- Eliminar el elemento mínimo
- Consultar el elemento mínimo
- Consultar si la cola de prioridad está vacía
- Consultar el tamaño de la cola de prioridad

La representación de la vista usuario de la herramienta será un conjunto de cajas ordenadas según hayan sido añadidas por el usuario, sobre las cuales se señalizará el elemento mínimo con una flecha.

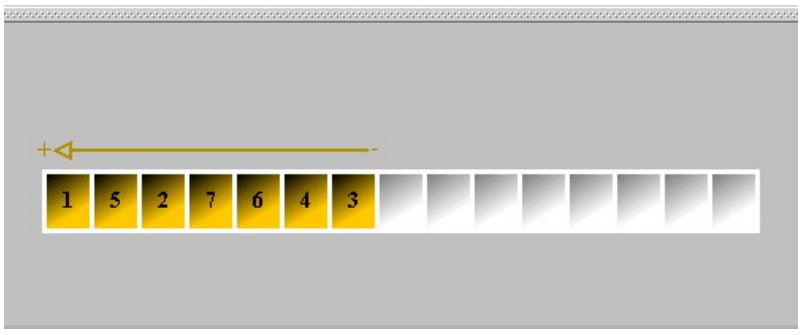
En la vista usuario de desarrollo con implementación estática se visualizará el montículo como un vector ordenado de más a menos prioridad.



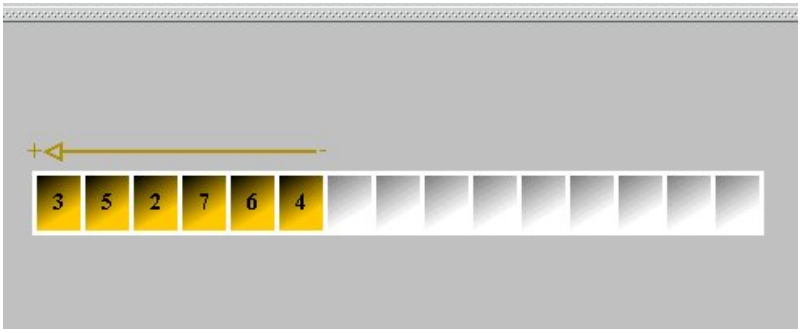
Los intercambios entre elementos producidos por durante las inserciones y eliminaciones por las operaciones de hundir y flotar respectivamente se indicarán mediante flechas.

A continuación se muestran el conjunto de pasos de animación necesario para eliminar el elemento que ocupa mínimo (en este caso el 1) a partir del montículo de la figura A. Como se puede observar tras la eliminación del elemento 1 se hace necesario realizar un hundimiento del elemento 3 ya que su hijo es más pequeño que él. En la figura C se muestra cómo se realiza tal intercambio y finamente en la figura D se señala cual es la posición que pasa a ocupar el elemento hundido.

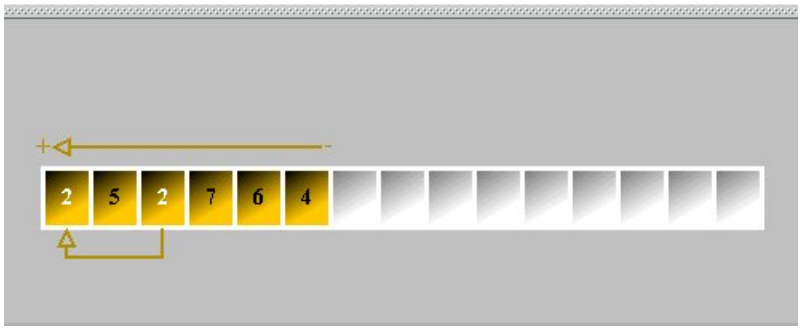
A



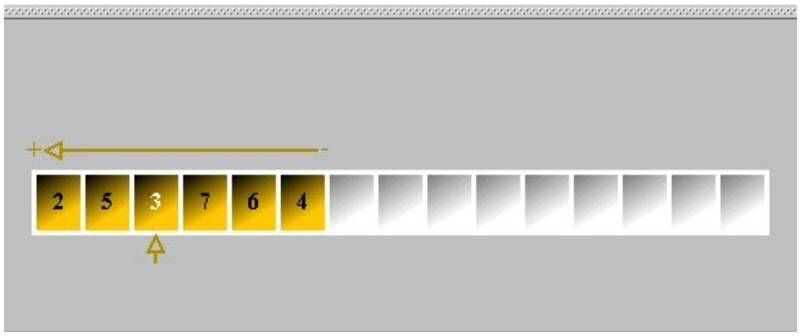
B



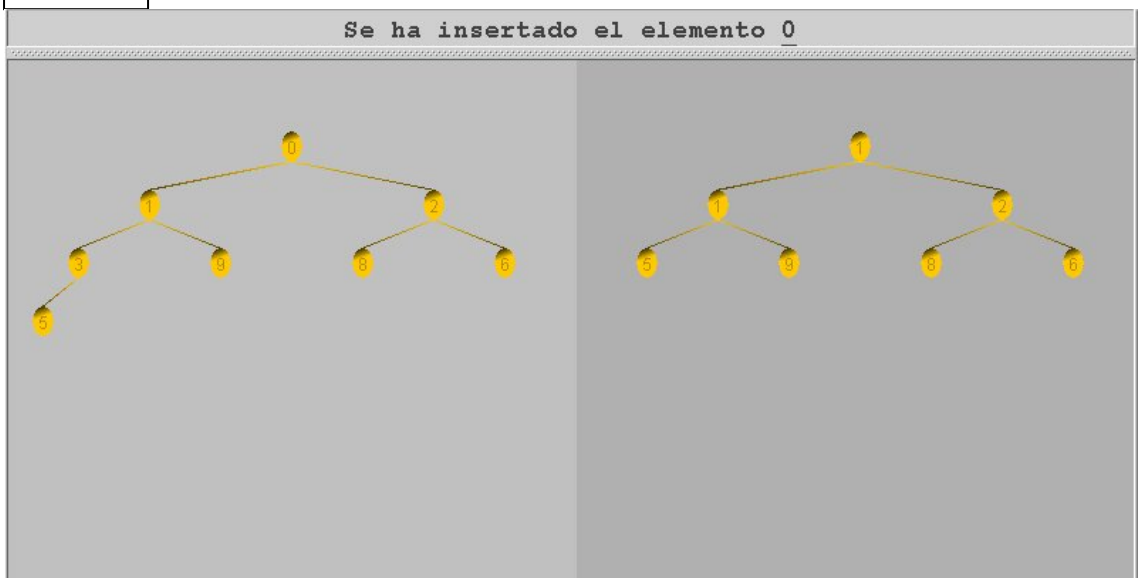
C



D



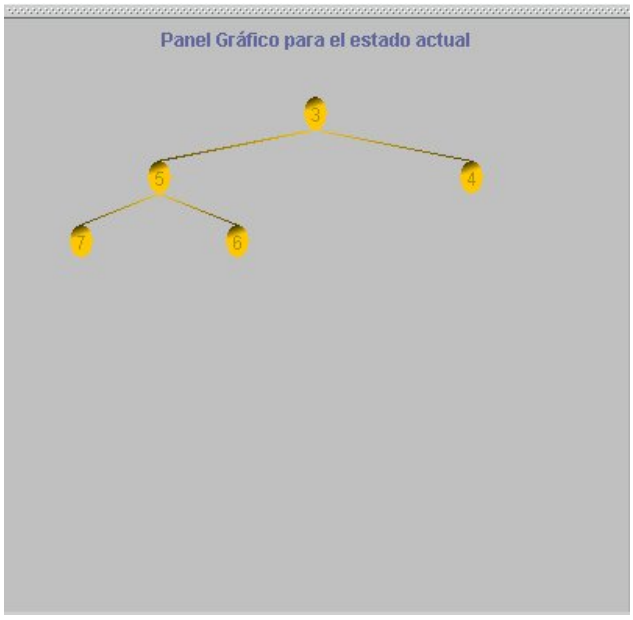
La estructura de datos montículo no tiene una implementación en forma de árbol, sin embargo la forma más natural de representar visualmente una cola de prioridad es mediante una estructura jerárquica en forma de árbol. Esta es otra de las posibilidades de representación que ofrece la herramienta.



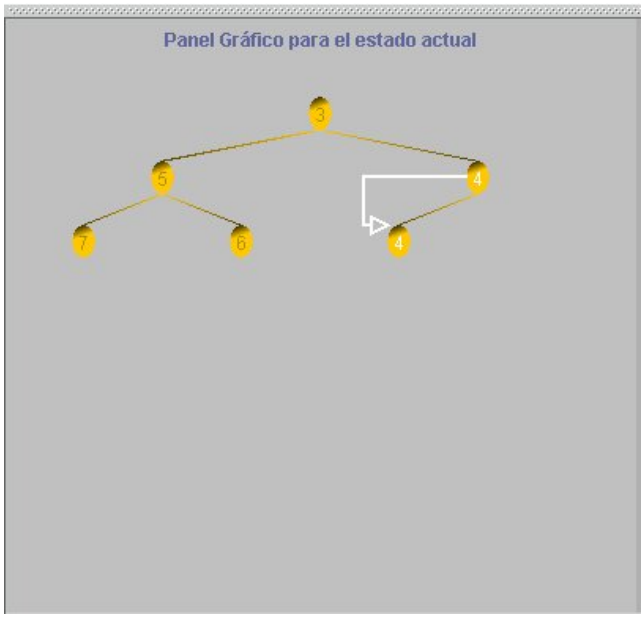
Al igual que en la vista estática los intercambios entre elementos se representan mediante flechas.

En la imagen que se muestra a continuación se puede observar los pasos realizados para insertar el elemento 2 a partir de la imagen A

A

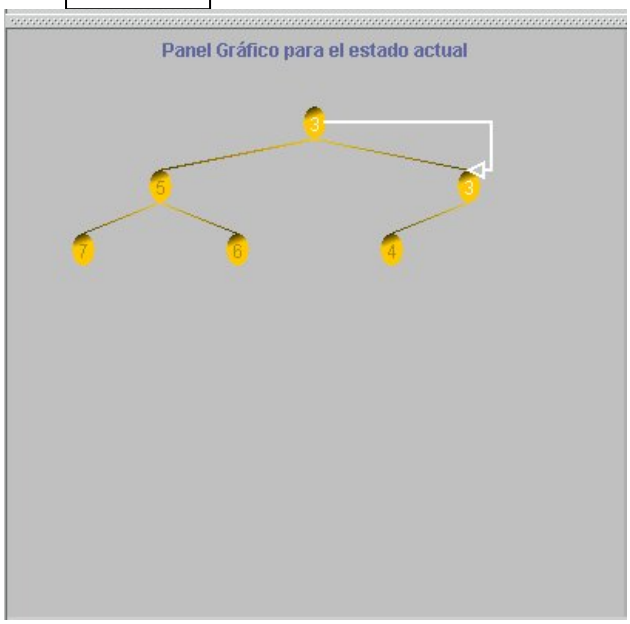


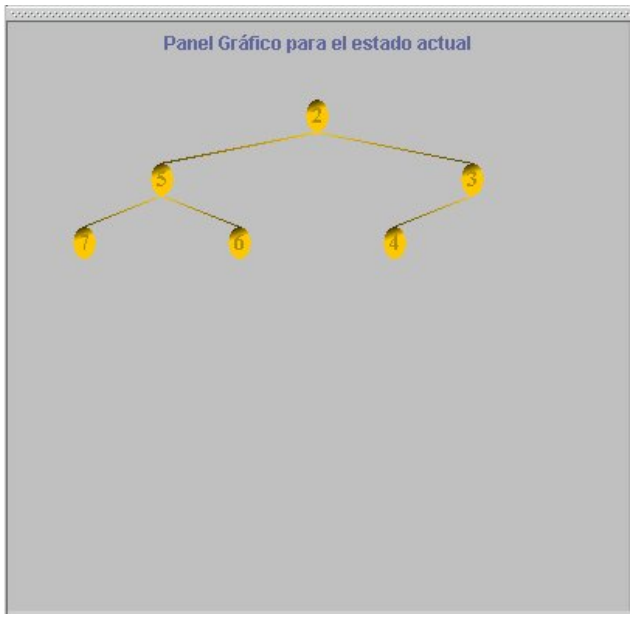
B



C

D





Como se puede observar lo primero que se hace antes de realizar la inserción es hacer sitio al elemento. Esto es equivalente a insertar primero el elemento y luego flotarlo. Ya que se sustituye el padre por el hijo hasta conseguir colocar el elemento.

2.1.4. Estructura de datos Árbol Binario de Búsqueda

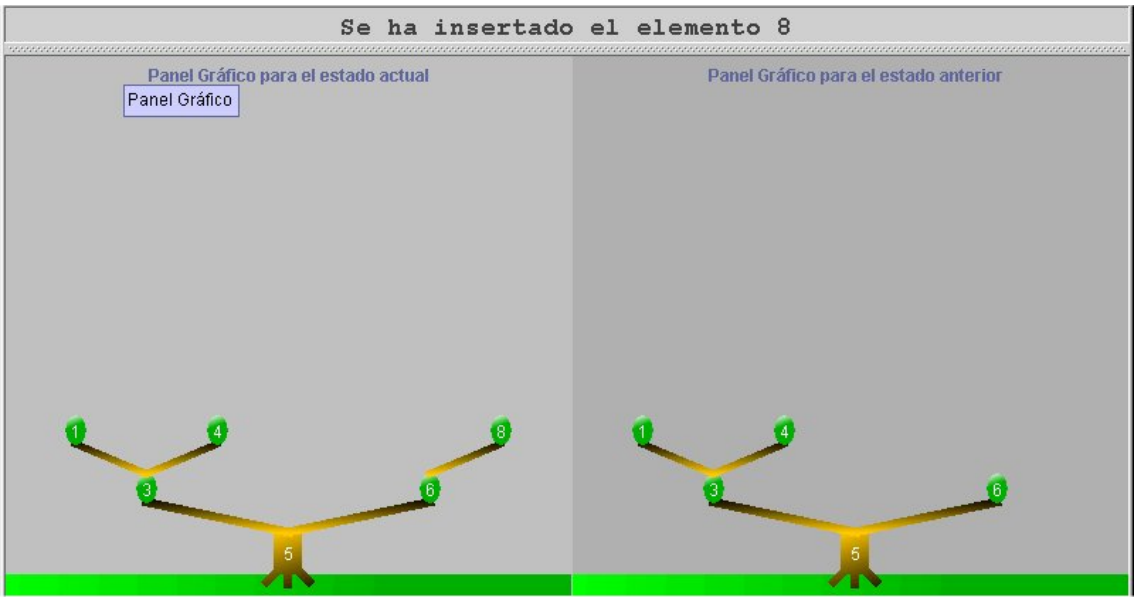
Sobre la estructura de datos árbol binario de búsqueda podrán aplicarse las siguientes operaciones:

- Crear
- Insertar
- Eliminar
- Realizar un recorrido en preorden del árbol
- Realizar un recorrido en recorrido postorden del árbol
- Realizar un recorrido en recorrido inorden del árbol
- Consultar los elementos de un determinado nivel
- Consultar la altura del árbol
- Consultar el hijo izquierdo del árbol
- Consultar el hijo derecho del árbol
- Consultar la raíz del árbol
- Consultar si la estructura arbórea está vacía

Los intercambios entre elementos producidos durante la eliminación se representarán mediante flechas.

En la vista usuario de la herramienta la representación de la estructura será la de un árbol semejante a los que se dan en la naturaleza, formado por una raíz, que se situará en la parte inferior del panel y ramas y hojas, los cuales crecerán de abajo a arriba.

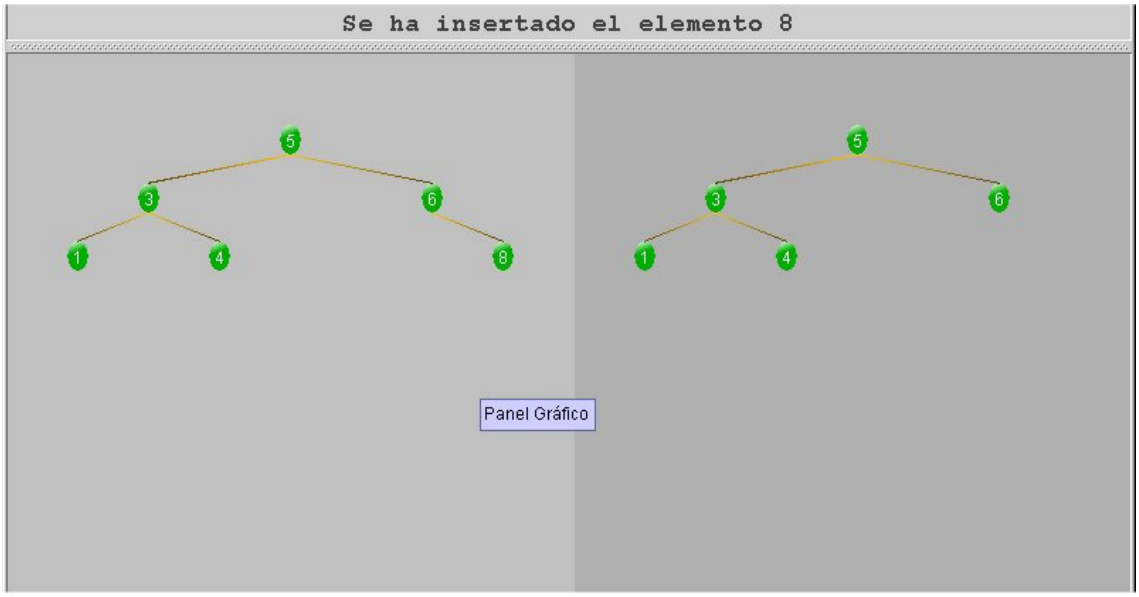
El árbol vacío se representará mediante un árbol carente de hojas.



En la vista de usuario de desarrollo con implementación estática se representará el árbol mediante un vector en el cual cada casilla representa una posición de un árbol completo recorrido en anchura.

En la vista de usuario de desarrollo con implementación dinámica basada en punteros: el árbol se representará con nodos dispuestos jerárquicamente, enlazados con punteros, de modo que cada nodo tiene dos hijos, uno a la izquierda y otro a la derecha.

Esta representación se ha desarrollado únicamente a efectos didácticos para que el alumno pueda observar el desperdicio de memoria que supone.

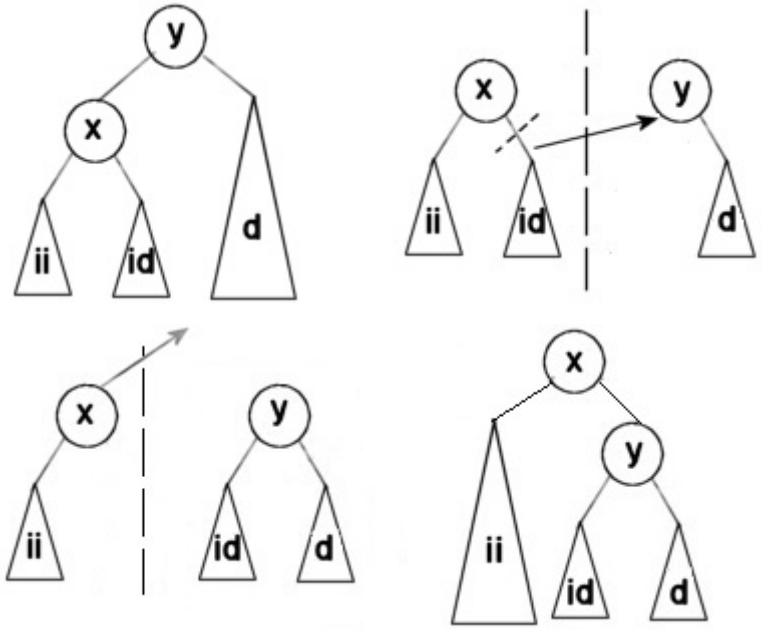


2.1.5. Estructura de datos Árbol AVL

Sobre la estructura de datos árbol AVL podrán aplicarse las misma operaciones que con el árbol binario de búsqueda. La estructura de datos AVL carecerá de vista de usuario de herramienta con implementación estática ya que no tiene sentido implementar un árbol AVL

con un vector. La representación del árbol en el resto de las vistas será equivalente a la del árbol binario de búsqueda.

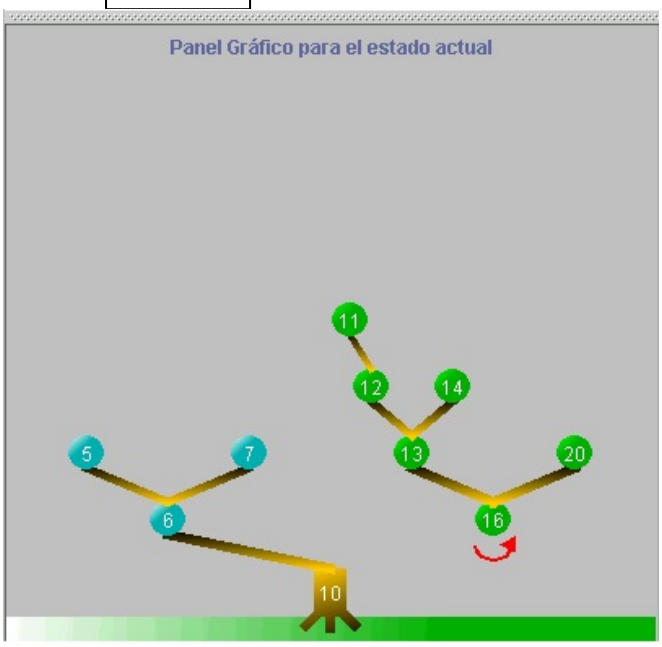
Las rotaciones se realizarán en varios pasos. Por ejemplo, si nos encontramos ante una rotación simple a la izquierda se realizarán los siguientes pasos.

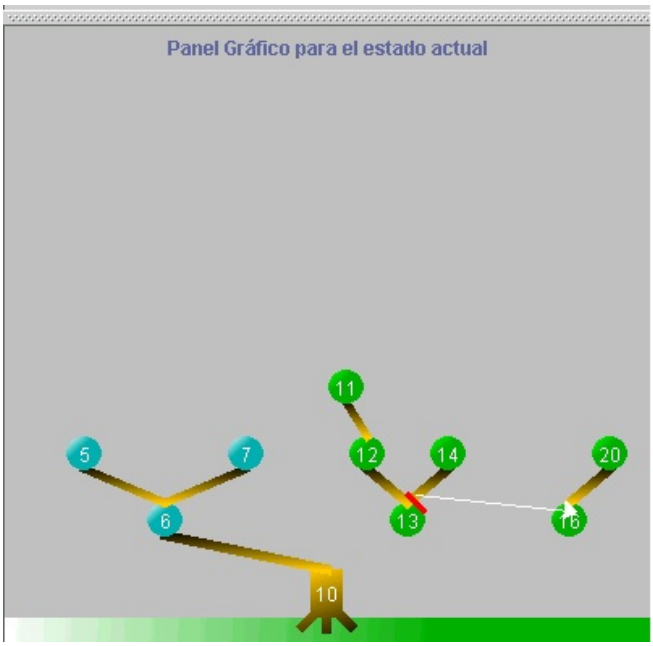


El tipo de rotación (izquierda o derecha) se representará mediante una flecha de color rojo que indica el sentido de rotación.

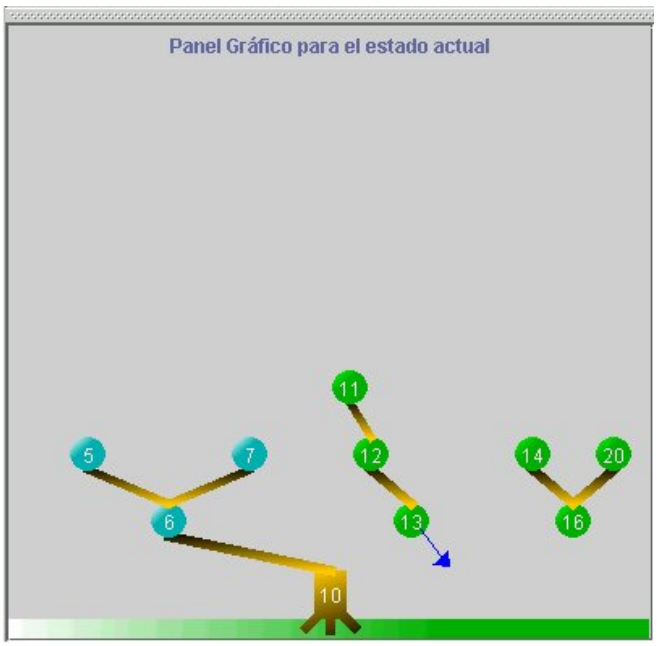
Si una rama perteneciente a un árbol requiere ser movida a otro árbol, se representará mediante una línea que corte la rama del árbol de partida y una flecha blanca que represente de dónde a donde se moverá la rama cortada.

Cuando un subárbol requiera bajar o subir para posicionarse se indicará mediante una flecha azul





Cuando un subárbol requiera bajar o subir para posicionarse se indicará mediante una flecha azul.



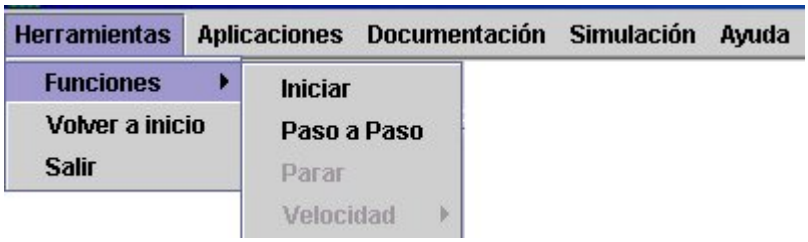
2.2. Esquemas algorítmicos

A diferencia de las estructuras de datos los esquemas algorítmicos no tienen una interfaz predefinida ya que ésta depende del algoritmo a desarrollar.

El menú de los algoritmos consta de las siguientes apartados:

- Herramientas

- Funciones:
  - Iniciar: Permite comenzar la visualización y animación del problema elegido.
  - Paso a Paso: Permite la visualización paso a paso de la ejecución del algoritmo que se esté ejecutando.
  - Parar: Interrumpe la ejecución del problema que se está realizando en ese momento.
  - Velocidad: Permite modificar la velocidad de ejecución del problema elegido.



  - Volver a Inicio: Permite volver a la pantalla de selección
  - Salir: Sale de la aplicación
- Aplicaciones: Permite seleccionar el problema con el que se desea trabajar
- Documentación:
  - Código: Contiene el código del algoritmo seleccionado por el usuario.
  - Costes
  - Ayuda Adicional: Información acerca del algoritmo seleccionado por el usuario.

- Simulación: Permite la ejecución de una simulación del problema elegido por el usuario.
- Ayuda:
  - Índice de materias: Contiene un catálogo de información que contiene todos los temas de ayuda disponibles, para facilitar el entendimiento y posibilitar la consulta, de cualquiera de las estructuras de datos y esquemas algorítmicos utilizados en la aplicación.
  - Manual de usuario: Ayuda acerca del uso de la herramienta

Durante la ejecución de los problemas, el usuario también podrá realizar las siguientes operaciones:

- Pausar: Una vez iniciada la ejecución se puede pulsar la opción pausar, la cual detiene momentáneamente la ejecución de la aplicación hasta el momento que lo desee el usuario. La opción pausar se sitúa en el lugar que ocupaba la opción iniciar. Si pulsa Ejecutar podrá continuar la ejecución del problema por el punto donde la suspendió.

- Ejecutar: ejecuta el ejemplo escogido desde el punto donde se detuvo al pausarse o desde el último punto que se ejecutó paso a paso.

Las funciones descritas en el menú de herramientas también se encontrarán disponibles en los botones que se encuentran en la interfaz, sobre la barra de estado.

2.2.1. Esquema Algorítmico Divide y Vencerás

Se dispone de dos problemas típicos que siguen la metodología divide y vencerás:

- Quicksort
- Búsqueda Binaria

Para ejecutar uno de estos dos problemas, una vez se encuentre situado en la ventana de visualización y animación del esquema algorítmico divide y vencerás, seleccione en el menú aplicaciones el problema con el que desee trabajar.

Búsqueda Binaria

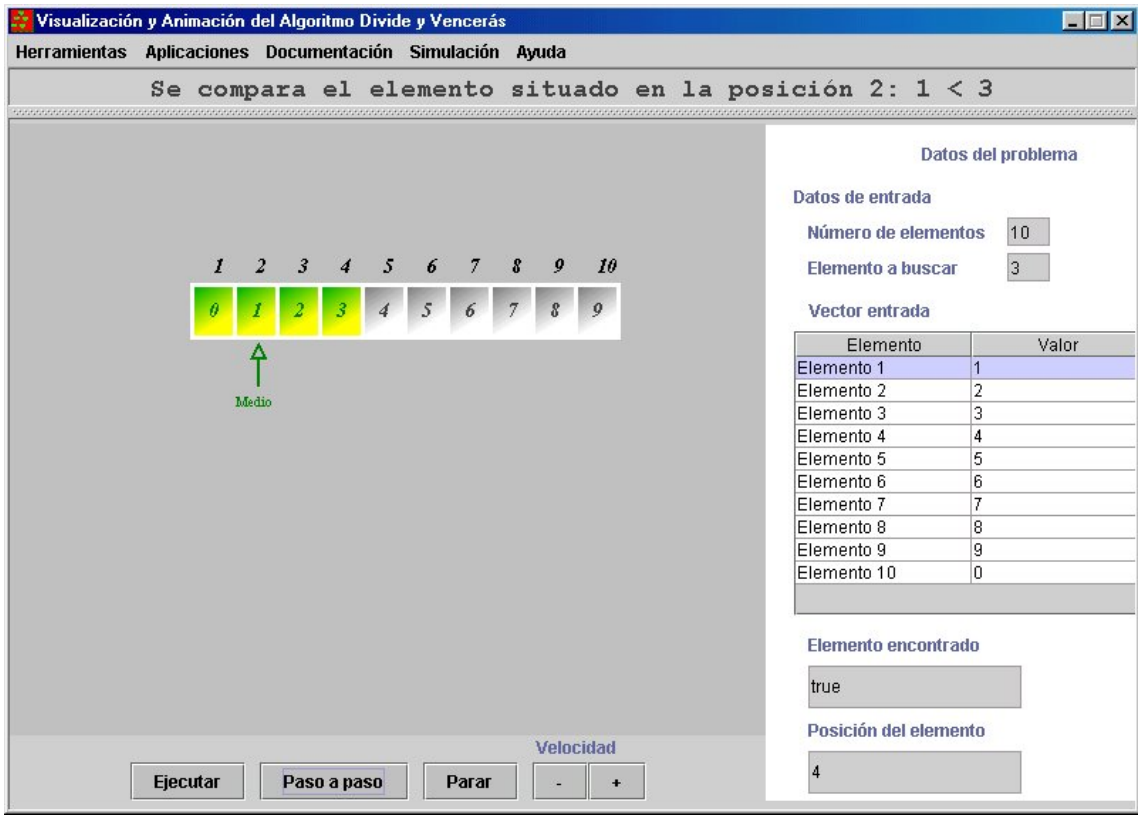
Introduzca los datos del problema:

- Número de elementos del vector
- Elemento a buscar
- Valor de los elementos del vector

Primero aparecerá el dato número elementos del vector, una vez que lo haya rellenado pulse INTRO, inmediatamente aparecerá el cuadro de texto *Elemento a buscar*, efectúe el mismo procedimiento que en el caso anterior (rellene y pulse INTRO) y aparecerá una tabla para que rellene el contenido del vector. Una vez haya rellenado todos los campos pulse INTRO, ello será imprescindible para que la aplicación recoja toda la información y verifique la aceptación de los datos que ha rellenado.

Pulse el botón *Iniciar* o *Paso a paso* para iniciar la ejecución del problema: Aparecerá un vector ordenado con los datos que introduzco inicialmente. La parte que se está procesando del vector aparecerá en color verde, mientras que el resto aparecerá en blanco y negro. Sobre el vector aparecerá la posición que ocupa cada elemento del vector. Se indicará con una flecha cual es el punto medio de la parte que se está procesando.

En la parte derecha, bajo la tabla de los datos del problema, aparecerá la solución del mismo, es decir si se ha encontrado el elemento y la posición que ocupa el mismo. Si el elemento buscado no se encuentra en el vector, se devolverá la posición que debería ocupar.



Quicksort

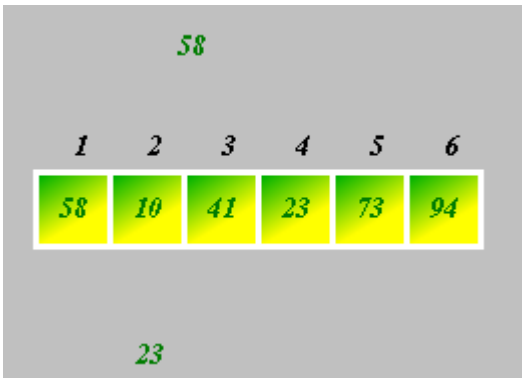
Introduzca los datos del problema:

- Número de elementos del vector
- Valor de los elementos del vector a ordenar

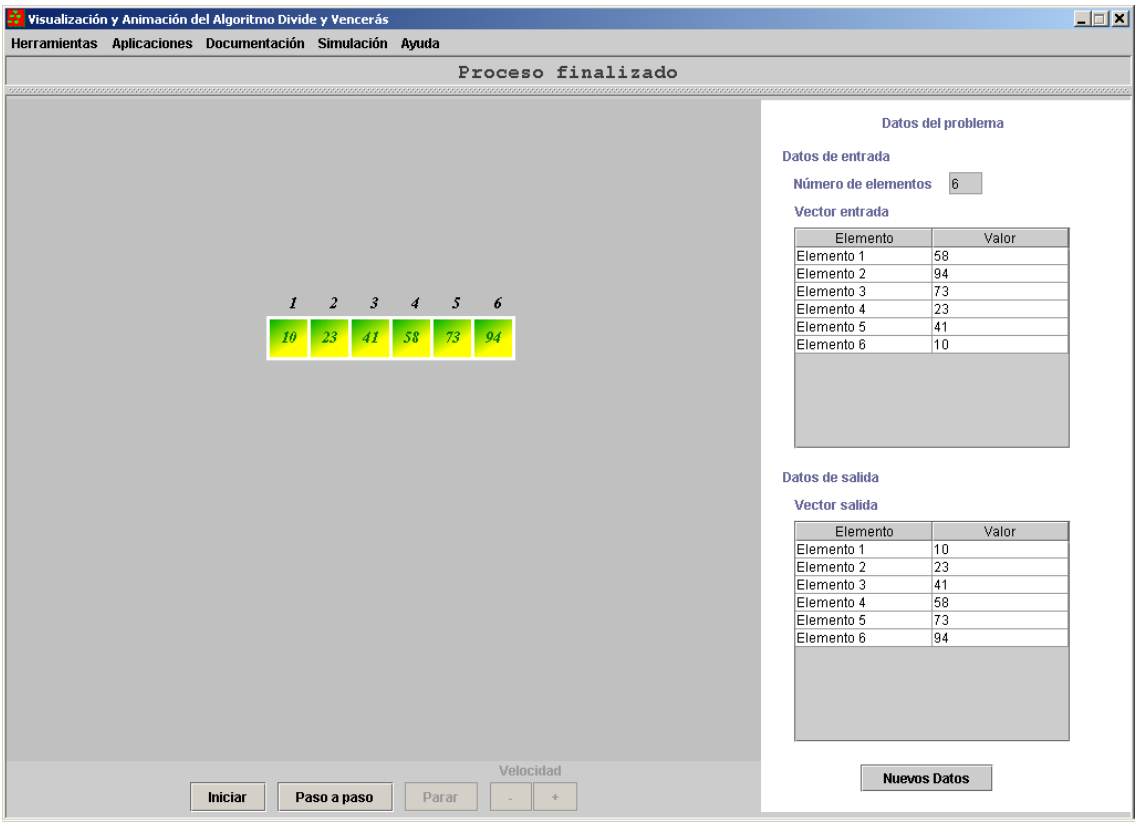
Primero aparecerá el dato número elementos del vector, una vez que lo haya rellenado pulse INTRO, inmediatamente aparecerá una tabla donde rellenar el contenido del vector. Una vez haya rellenado todos los campos del vector pulse INTRO, ello será imprescindible para que la aplicación recoja toda la información y verifique la aceptación de los datos que ha rellenado.

Pulse el botón *Iniciar* o *Paso a paso* para iniciar la ejecución del problema: Aparecerá un vector desordenado con los datos que ha introducido. La parte que se está procesando del vector aparecerá en color, mientras que el resto aparecerá en blanco y negro. Sobre el vector aparecerá la posición que ocupa cada elemento del vector. Se indicará con una

flecha cual es la posición del pivote y cual es la posición de los índices derecho e izquierdo que recorren el vector. Los elementos que se intercambien saldrán uno por la parte de arriba del vector y el otro por la parte de abajo del vector.



En la parte derecha bajo la tabla de los datos del problema aparecerá la solución del mismo, es decir el vector ordenado.



2.2.2. Esquema Algorítmico Voraz

Los problemas que se proporcionan como ejemplos voraces son:

- Problema de la mochila
- Algoritmo de Dijkstra

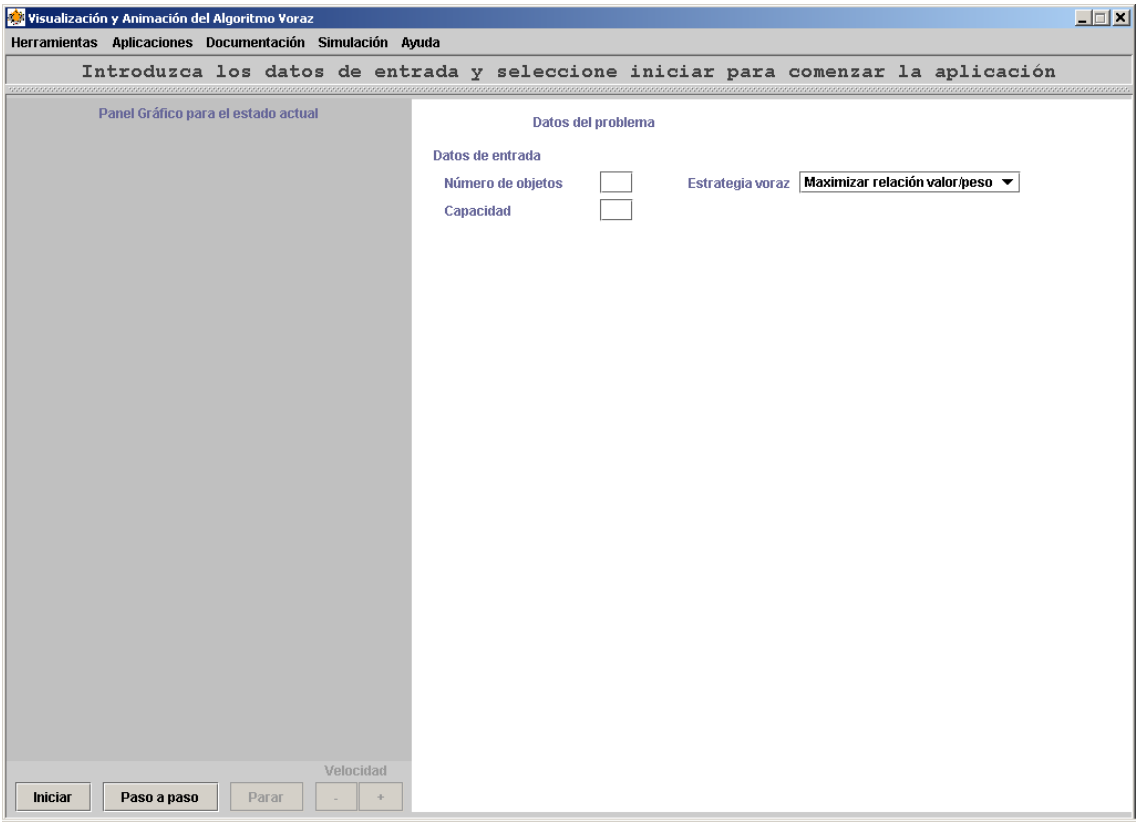
Problema de la mochila

Al seleccionar este problema aparecerá una pantalla en la que se distinguen dos zonas claramente diferenciadas:

- A la izquierda se encuentra la zona donde se representará gráficamente la ejecución del problema.
- A la derecha se encuentra la zona dónde se rellenarán los datos del problema y la solución del mismo.

Para iniciar la ejecución del problema rellene los datos de entrada:

- Estrategia voraz
  - Maximizar peso
  - Maximizar valor
  - Maximizar relación valor/peso
- Número de objetos que podrán formar parte de la mochila
- Capacidad máxima de la mochila



Una vez rellene la capacidad de la mochila pulse INTRO, ello será imprescindible para que la aplicación recoja toda la información y verifique la aceptación de los datos introducidos. Tras ello aparecerá una tabla que contiene para cada objeto su valor, peso y relación valor/peso. Rellene únicamente las columnas valor y peso, recordando que tras rellenar cada celda editable se debe pulsar un INTRO para la confirmación de los datos introducidos. Una vez haya rellenado estas dos columnas, pulse *Iniciar* o *Paso a paso* para empezar la ejecución del algoritmo.

Al iniciar la ejecución del problema, en la tabla vector de entrada se rellenará la relación valor peso y aparecerán los siguientes colores en las celdas, los cuales indican:

- Las celdas con fondo rosa indican los elementos que se estám procesando.
- Las celdas con fondo de color turquesa son los objetos seleccionados.

Veamos el siguiente fragmento de un problema donde se está realizando una estrategia de maximización relación valor/peso

Datos de entrada

Número de objetos	5
Capacidad	10

Vector entrada

Objeto	Valor	Peso	Valor/Peso
Objeto 1	5	2	2,50
Objeto 2	2	3	0,67
Objeto 3	4	4	1,00
Objeto 4	3	5	0,60
Objeto 5	2	6	0,33

Vector entrada

Objeto	Valor	Peso	Valor/Peso
Objeto 1	5	2	2,50
Objeto 2	2	3	0,67
Objeto 3	4	4	1,00
Objeto 4	3	5	0,60
Objeto 5	2	6	0,33

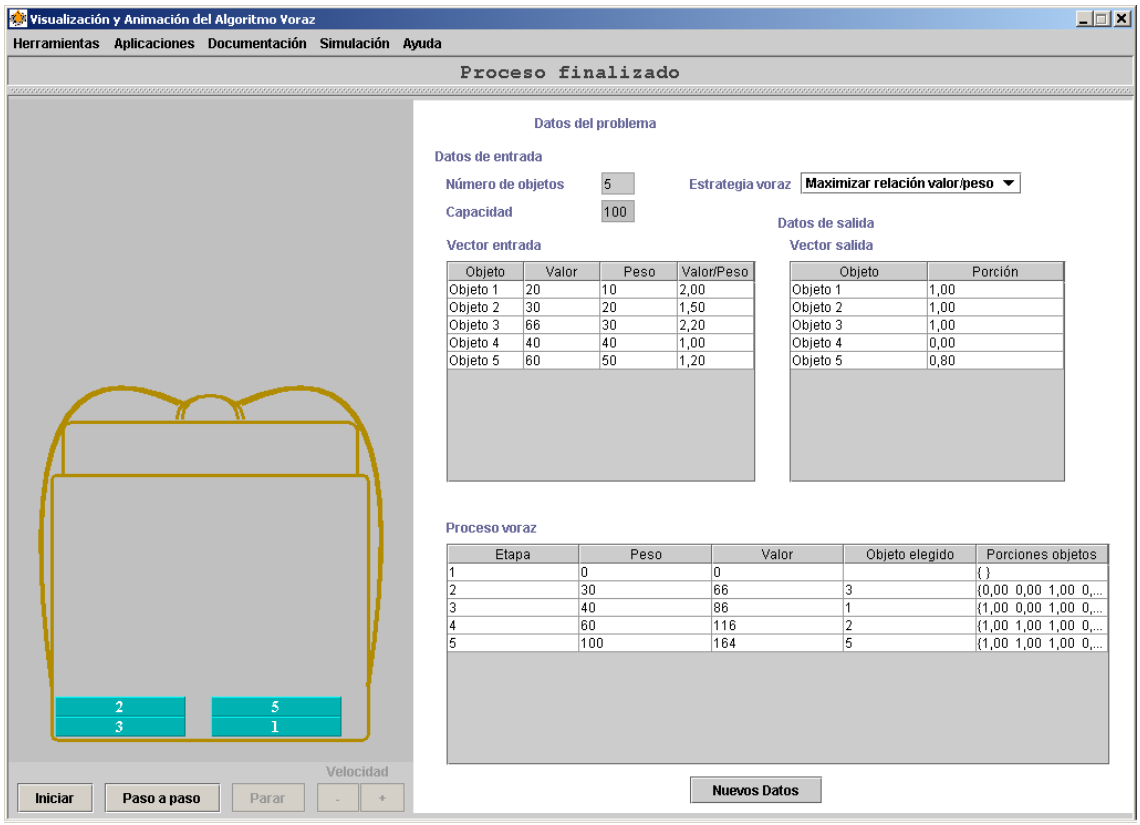
Vector entrada

Objeto	Valor	Peso	Valor/Peso
Objeto 1	5	2	2,50
Objeto 2	2	3	0,67
Objeto 3	4	4	1,00
Objeto 4	3	5	0,60
Objeto 5	2	6	0,33

Como se puede observar inicialmente, en la primera etapa del proceso voraz, se seleccionan todos los objetos para ser procesados, posteriormente el objeto que se

selecciona entre los procesados es el que tiene mayor relación valor/peso, en nuestro ejemplo el objeto 1, el cual será introducido en la mochila.

En la parte inferior derecha de la pantalla aparecerá una tabla donde se irá completando el proceso voraz seguido en cada etapa del problema. Junto a la tabla vector de entrada aparecerá la tabla de datos de salida, la cual contiene para cada objeto la porción que ocupa en la mochila.



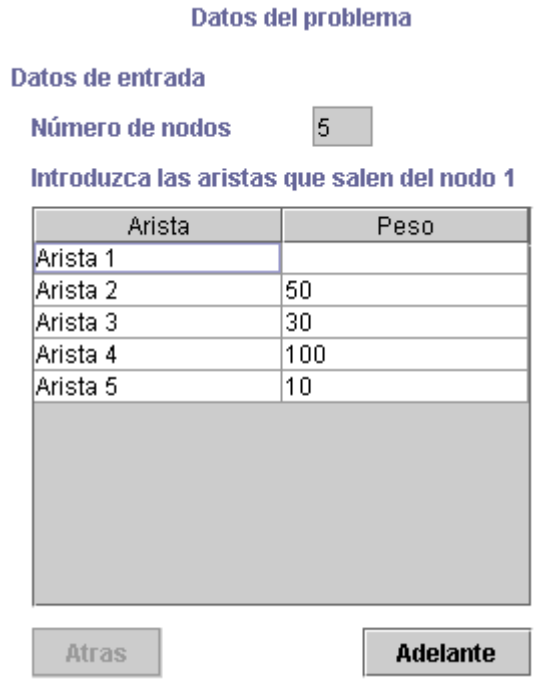
Algoritmo de Dijkstra

En la interfaz del problema de la mochila distinguiremos dos zonas claramente diferenciadas:

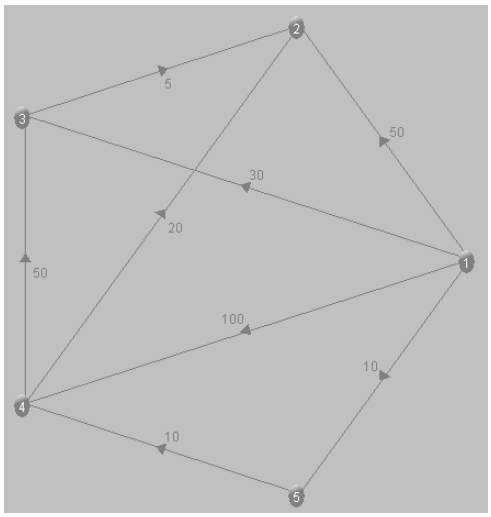
- Zona de representación del grafo.
- Zona de datos entrada del problema y solución del problema

Para empezar a trabajar deberá rellenar el dato referente al número de nodos del grafo, al menos se requerirán dos nodos. Pulse INTRO, si se introduce menos de dos nodos, se mostrará un mensaje de error que informe del hecho. Si el número de nodos es adecuado, se mostrará una tabla donde rellenar las aristas existentes del grafo que salen del nodo 1 al resto de los nodos. Las celdas rellenadas representarán el peso de las aristas, mientras que las celdas vacías indicarán que no sale ninguna arista del nodo 1 al nodo i (celda que se

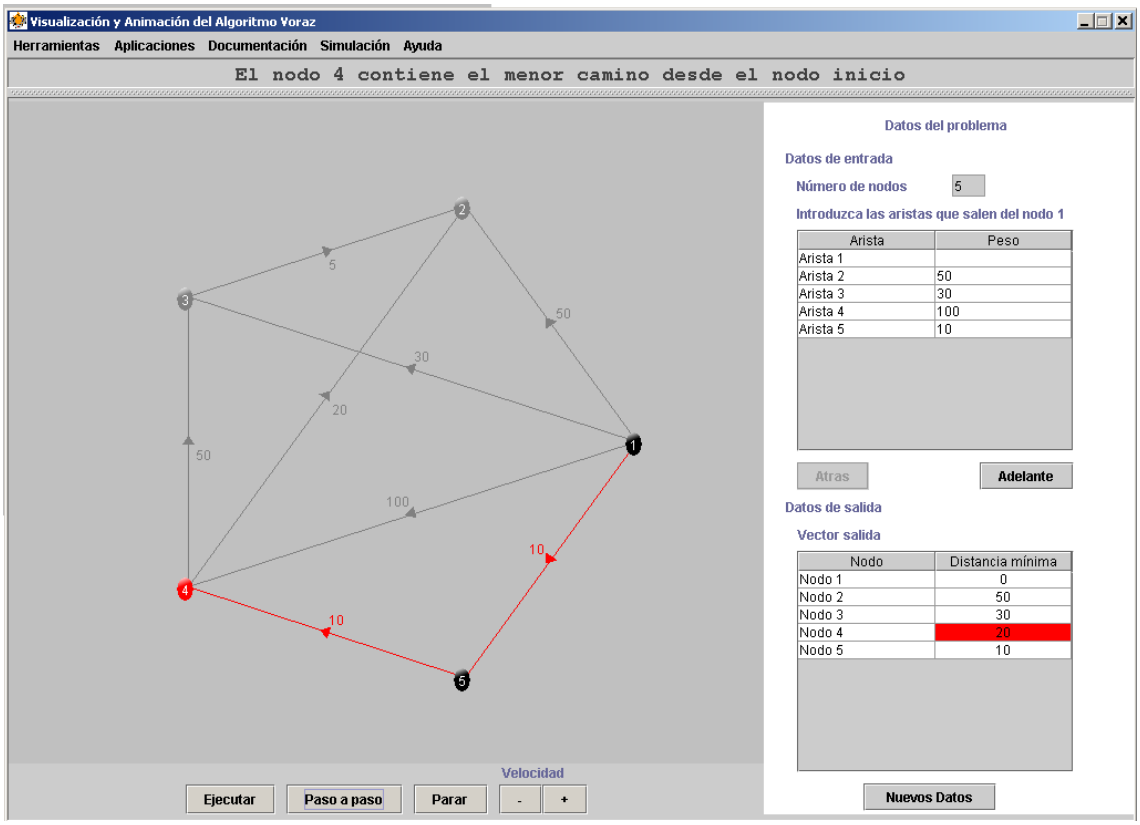
encuentra en la fila i). Una vez rellenado las aristas que salen del nodo 1, se podrá rellenar las aristas que salen del resto de los nodos, pulsando el botón *Adelante*. En todo momento podrá modificar las aristas rellenadas con anterioridad pulsando el botón *Atrás*. No olvide pulsar el INTRO tras insertar, eliminar o modificar el valor de cada celda. Cuando se encuentre en la última tabla de aristas, correspondiente al último nodo del grafo, se deshabilitará el botón *Adelante*. El botón *Atrás* se encontrará deshabilitado en la tabla de aristas del nodo 1.



Tras la inserción de todos los datos pulse *Iniciar* o *Paso a paso* para iniciar la ejecución del algoritmo. Si algunos de los datos no son correctos, se mostrará un mensaje de error que indique los motivos posibles del fallo. Si los datos son correctos, aparecerá a la izquierda de la pantalla el grafo a tratar, y en la parte inferior derecha el vector de salida del problema, inicialmente vacío.



Durante la ejecución del algoritmo se mostrará en la etiqueta descriptiva, situada en la parte superior de la interfaz, las acciones llevadas a cabo. Sobre el grafo se visualizarán con distintos colores, según la operación ejecutada, los nodos y aristas implicadas. Sobre el vector de salida se colorearán las celdas tratadas o modificadas.



Las celdas de color amarillo indicarán los nodos no procesados, esto es, los nodos que hasta el momento no conocen su camino mínimo y que van a ser tratados o que pueden ser seleccionados. La celda de color rojo indicará el objeto seleccionado que conoce su camino

mínimo. Finalmente la celda de color verde reflejará los nodos cuya distancia mínima se verá modificada.

Al mismo tiempo que se colorean las celdas, se podrá visualizar sobre el grafo los nodos implicados. Los nodos de color negro representarán los nodos tratados, al igual que las aristas que forman parte de su camino mínimo. Durante la ejecución, los nodos y las aristas implicadas se podrán mostrar de color rojo o verde, cuyo significado es el mismo que el explicado en las celdas.

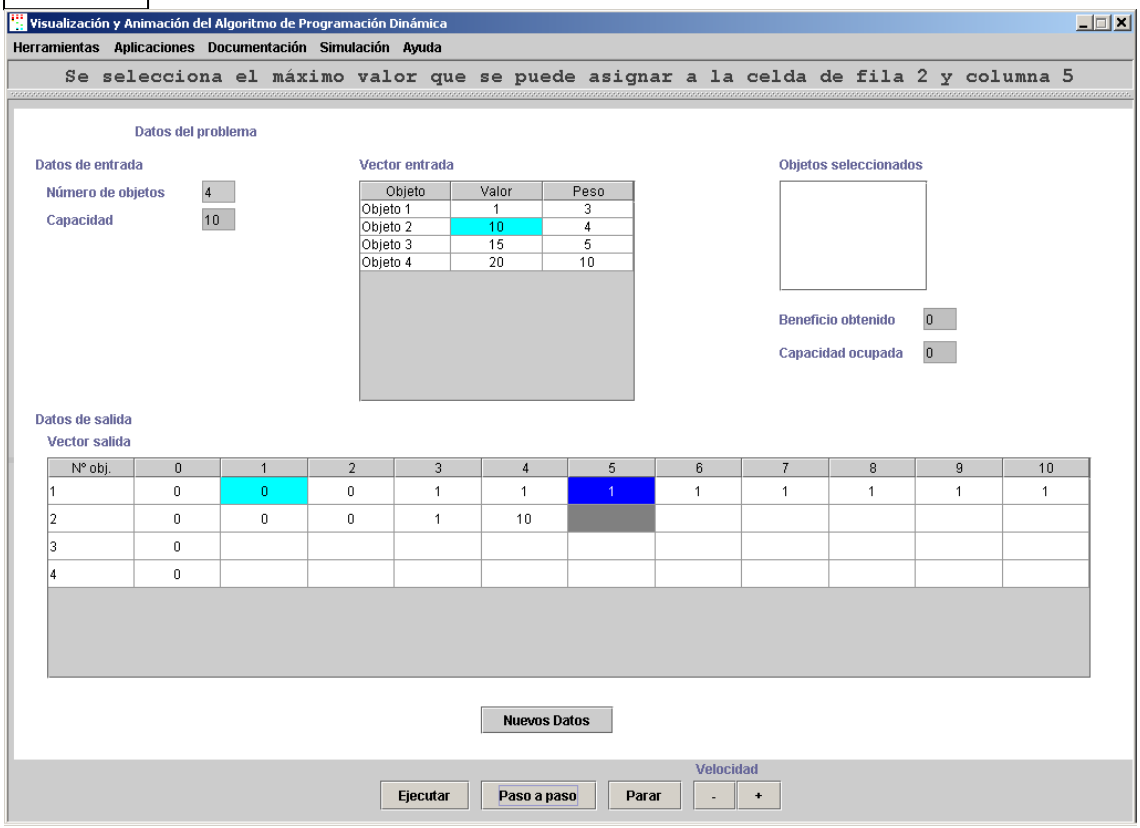
2.2.3. Esquema Algorítmico Programación Dinámica

El problema que se proporciona en el esquema algorítmico de programación dinámica es el problema de la mochila.

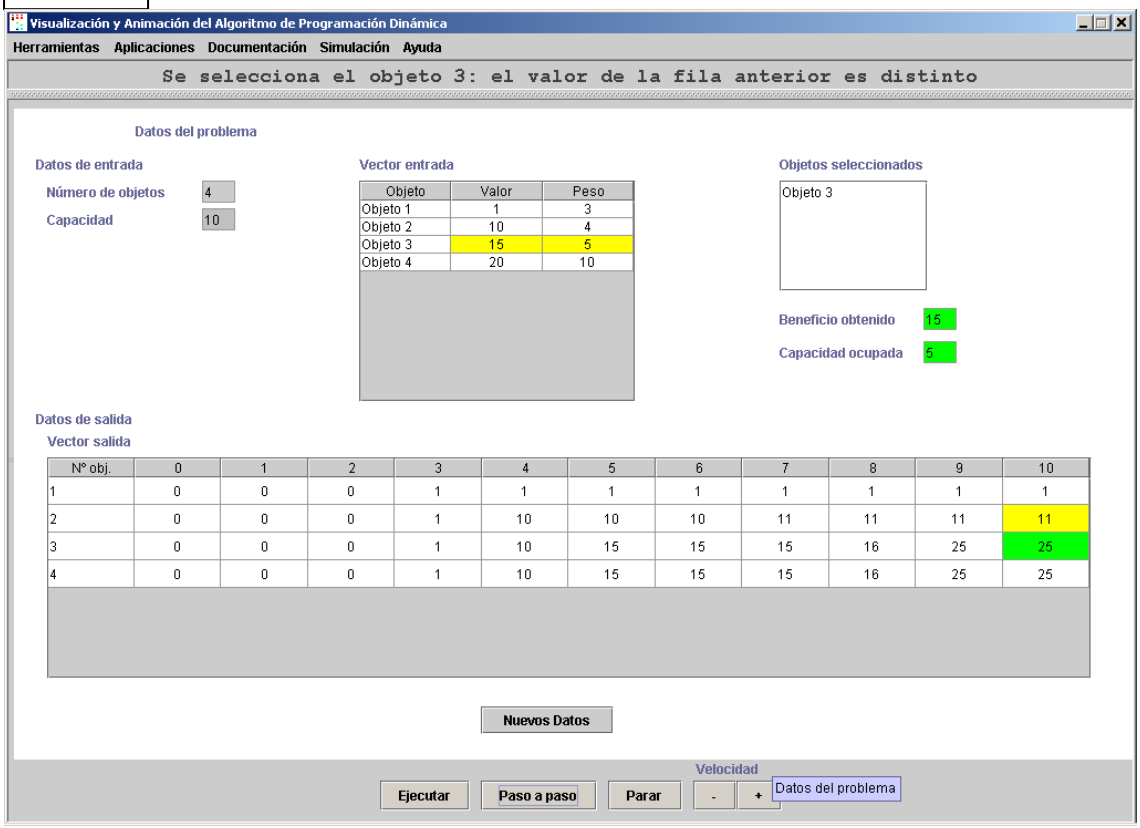
Problema de la mochila

Para empezar a trabajar con este problema, deberá completar los datos de entrada del problema. Rellene los campos número de objetos y capacidad, tras ello pulse INTRO para confirmar. Una vez hecho esto, se visualizará una tabla donde se deberá rellenar para cada objeto su valor y peso, pulsando INTRO tras la introducción de cada uno de ellos para la confirmación de los datos introducidos. Pulse *Iniciar* o *Paso a paso* para iniciar la ejecución del problema. En la tabla de datos de salida aparecerán los siguientes colores, los cuales indicarán:

- Fase de construcción de la tabla:
  - Las celdas coloreadas en color gris oscuro indican las casillas que se están procesando.
  - Las celdas coloreadas en color turquesa y azul oscuro indican las casillas que forman parte de la operación de selección del máximo valor.
  - Las celdas coloreadas en color rosa serán las casillas implicadas en la asignación del valor de la celda visualizada en color gris oscuro.



- Fase de reconstrucción de la solución:
  - Las celdas o cuadros de texto de color amarillo representarán los componentes utilizados para decidir si el objeto i forma parte de la mochila.
  - Las celdas de color verde de la matriz indicarán el objeto i que se esta procesando, los cuadros de texto se mostrarán de color verde cuando se modifique el valor del beneficio y capacidad ocupada de la mochila.



En la lista se reflejará los objetos insertados en la mochila.

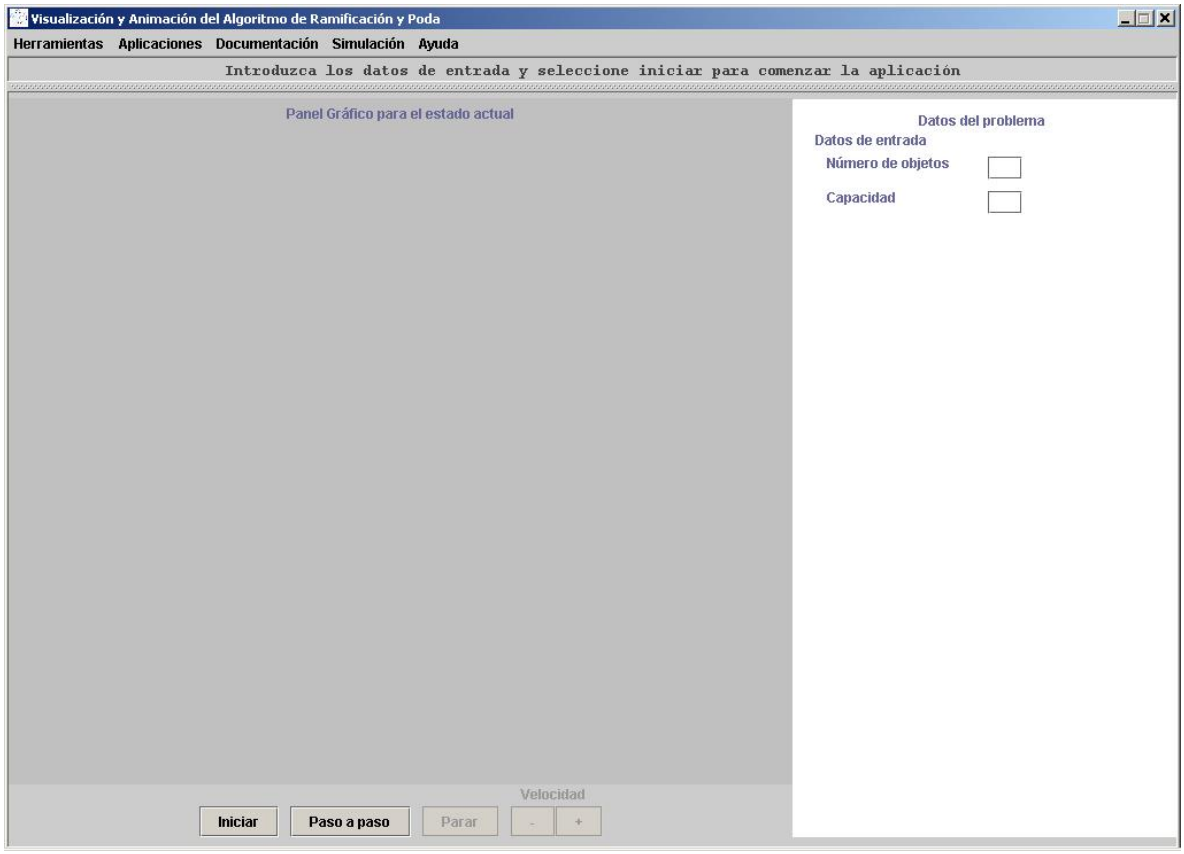
2.2.4. Esquema Algorítmico Ramificación y Poda

El problema que se proporciona como ejemplo de programación dinámica es el problema de la mochila

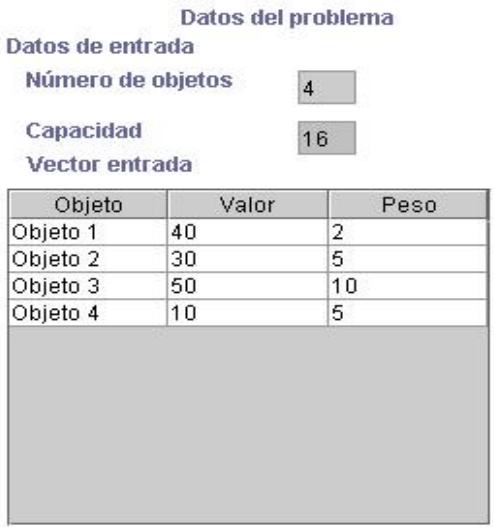
Problema de la mochila

Al seleccionar este problema aparecerá una pantalla en la que se distinguen dos zonas claramente diferenciadas:

- A la izquierda se encuentra la zona donde se representará gráficamente la ejecución del problema: representándose en la zona superior el árbol construido y en la zona inferior la cola de prioridad correspondiente.
- A la derecha se encuentra la zona dónde se rellenarán los datos del problema y la solución del mismo.

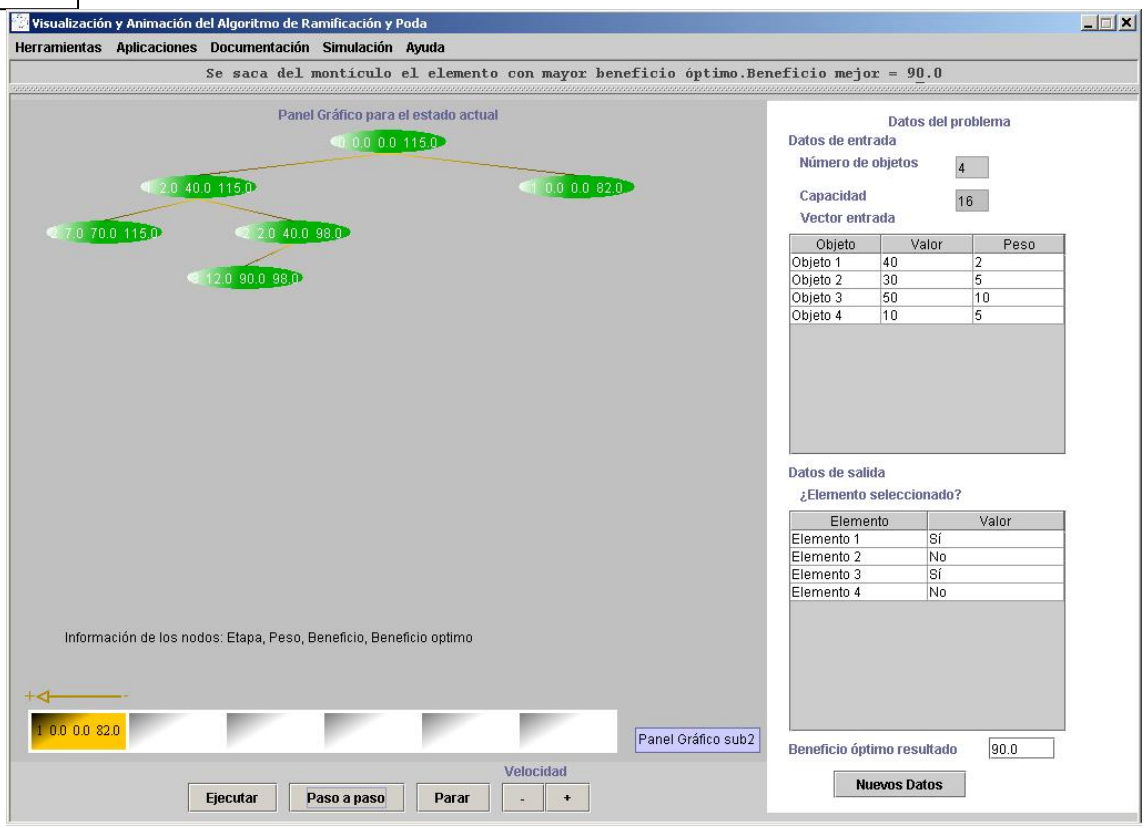


Rellene los datos de entrada del problema. Introduzca el número de objetos de la mochila y la capacidad de la misma. Pulse INTRO para la confirmación de los datos introducidos, inmediatamente aparecerá una tabla en la que se deberá rellenar para cada objeto su valor y su peso.



Pulse iniciar o paso a paso para empezar la ejecución del problema. Debajo de los datos de entrada aparecerá la solución del problema, la cual será una tabla donde se indicará para cada objeto si ha sido o no seleccionado, además del beneficio óptimo del nodo solución.

La información que contendrá cada nodo de la representación gráfica será la etapa, el peso, el beneficio y el beneficio óptimo.

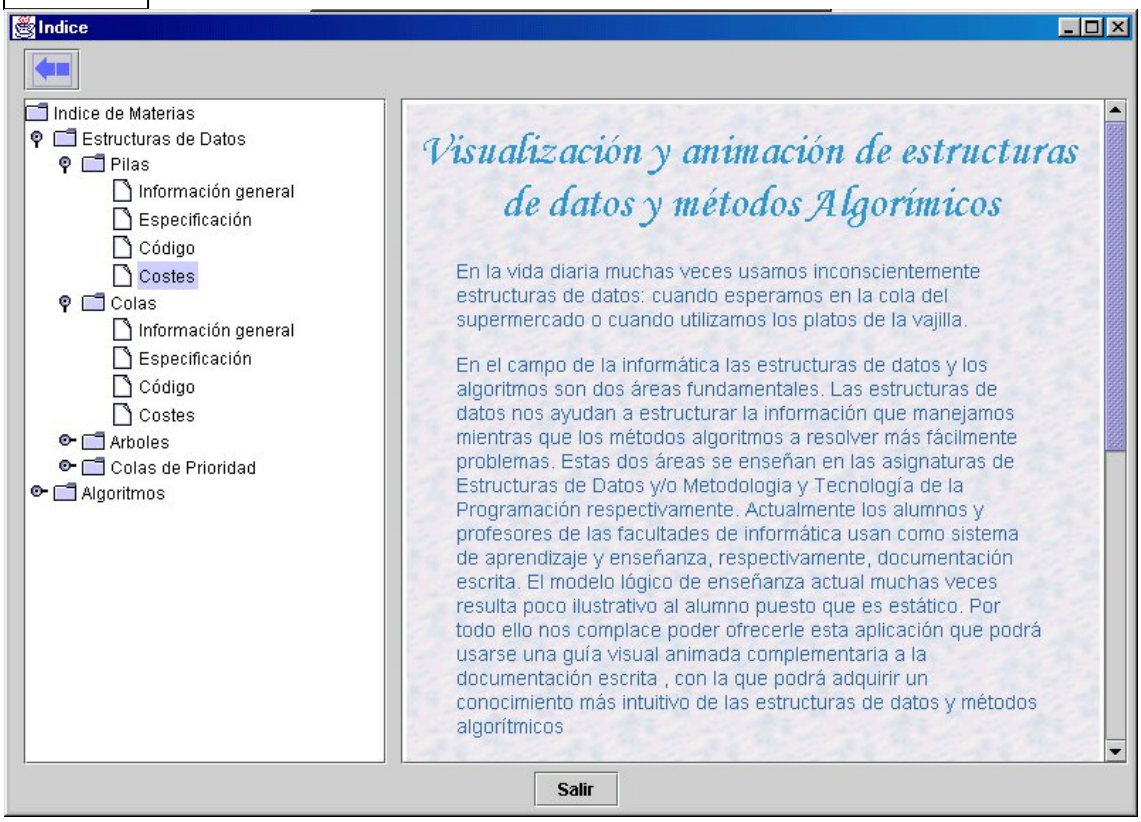


3. Ayuda

Desde el índice se podrá acceder a toda la ayuda disponible en el sistema. La pantalla de ayuda se divide en varias zonas. A la izquierda de la pantalla de ayuda aparecerá un índice de los temas de ayuda. Si el usuario selecciona alguno de los temas disponibles se mostrará su contenido el la zona derecha de la pantalla, es decir, aparecerá el tema de ayuda seleccionado.

Si no se ha seleccionado ningún tema de ayuda en concreto se mostrará una documento de las características generales de la herramienta.

En la parte superior izquierda se dispone de un botón que permite consultar páginas previamente consultadas.


Indice
Indice de Materias
Estructuras de Datos
Pilas
Información general
Especificación
Código
Costes
Colas
Información general
Especificación
Código
Costes
Arboles
Colas de Prioridad
Algoritmos
Visualización y animación de estructuras de datos y métodos Algorímicos
En la vida diaria muchas veces usamos inconscientemente estructuras de datos: cuando esperamos en la cola del supermercado o cuando utilizamos los platos de la vajilla.
En el campo de la informática las estructuras de datos y los algoritmos son dos áreas fundamentales. Las estructuras de datos nos ayudan a estructurar la información que manejamos mientras que los métodos algoritmos a resolver más fácilmente problemas. Estas dos áreas se enseñan en las asignaturas de Estructuras de Datos y/o Metodologia y Tecnología de la Programación respectivamente. Actualmente los alumnos y profesores de las facultades de informática usan como sistema de aprendizaje y enseñanza, respectivamente, documentación escrita. El modelo lógico de enseñanza actual muchas veces resulta poco ilustrativo al alumno puesto que es estático. Por todo ello nos complace poder ofrecerle esta aplicación que podrá usarse una guía visual animada complementaria a la documentación escrita , con la que podrá adquirir un conocimiento más intuitivo de las estructuras de datos y métodos algorítmicos
Salir

VALORACIÓN DEL TRABAJO REALIZADO

La herramienta desarrollada nos ha dado la oportunidad de conocer nuevas funcionalidades del lenguaje de programación Java, reforzar el conocimiento de las asignaturas de Estructuras de Datos y Esquemas Algorítmicos, enfrentarnos al desarrollo de una aplicación destinada a los alumnos.

De entre los conocimientos adquiridos en Java podemos destacar, la animación usando concurrencia, nuevas funcionalidades de Swing, tales como *JTree, JTable, JScrollBar* y *JScrollPane*, el dibujado mediante la clase Java2D, la visualización de páginas web, el manejo de iteradores, la implementación de los métodos de dibujado de árboles.

El sistema desarrollado es un sistema complementario al sistema de enseñanza actual, el cual permite visualizar de forma animada el comportamiento de las estructuras de datos y algoritmos implementados, facilitando el entendimiento de los alumnos y fomentando un aprendizaje más ameno y motivador.

Aunque el sistema no se ha desarrollado en su totalidad por falta de tiempo, no pudiéndose implementar estructuras de datos más complejas y otros algoritmos, consideramos que es una herramienta completa, útil y con facilidad de ampliación. El prototipo final obtenido permite reflejar todas las funcionalidades que podrá abarcar.

Apendice A: Java 2D

La clase *Graphics2D* hereda de *Graphics*. *Graphics2D* aporta a las funcionalidades de *Graphics* mayor control sobre las transformaciones de coordenadas y sobre el manejo del color, texto y formas geométricas.

Para dibujar una forma se han de seguir los siguientes pasos:

1. Obtener un objeto de la clase *Graphics2D*:

   Para usar la clase *Graphics2D* se debe hacer un molde del objeto *Graphics* obtenido del método *paintComponent(Graphics g)*:

   Graphics2D g2 = (Graphics2D)g;

2. Establecer los hints de representación para llegar a un compromiso entre velocidad de representación y calidad del dibujo. Para ello se usará:

   RenderingHints hints=....;

   g2.setRenderingHints(hints);

3. Establecer el trazo(grosor y continuidad de la línea)

   Stroke stroke=...;

   g2.setStroke(stroke);

4. Establecer el relleno:

   Paint paint=...;

   g2.setPaint(paint)

5. Establecer la región de recorte:

   Shape clip=...;

   g2.clip(clip);

6. Definir las formas en un sistema de coordenadas personal, es decir aplicar rotaciones y/o traslaciones al sistema de coordenadas actual para llegar a un nuevo sistema.

   AffineTransform transform=...;

   g2.transform(transform);

7. Definir reglas de composición para combinar nuevos pixeles con los existentes en nuestro dibujo

   Composite composite=....;

   g2.setComposite(composite);

8. Crear la forma:

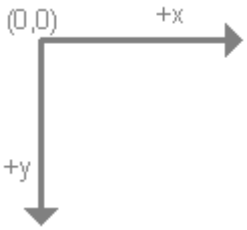
   Shape shape=...;

9. Dibujar y/o rellenar la forma:

   g2.draw(shape);

g2.fill(shape);

El sistema de coordenadas por defecto es el que se muestra a continuación:



Shapes

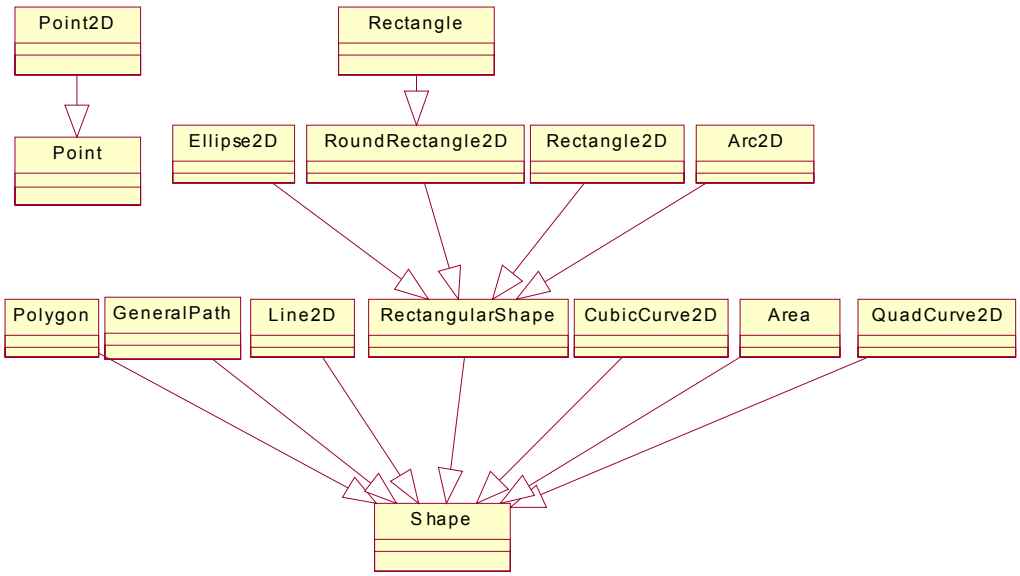
La jerarquía de clases con las que trabaja Java2D es la que se muestra a continuación.

Como podemos observar existe una clase *Shape* de la que heredan todas las formas geométricas excepto los puntos (*Point*).

La clase *Shape* contiene tanto formas geométricas simples, que son aquellas formadas por una sola forma geométrica como compuestas, que son aquellas formadas por composición de formas geométricas simples.

Las formas geométricas simples son los polígonos, líneas, formas rectangulares y las curvas cuadráticas y cúbicas. Mientras que las compuestas son las áreas y los *GeneralPath*.

Las formas rectangulares son los rectángulos, rectángulos de extremos redondeados, elipses y arcos. Se denominan así debido a que toman como referencia un rectángulo sobre el que se inscriben.



Métodos más significativos:

- Constructoras:
  - Rectangle2D rectangulo = Rectangle2D.Double (x, y, ancho, alto)
  - Ellipse2D elipse = Ellipse2D.Double(x, y, ancho, alto)
  - RoundRectangle2D rectanguloRedondeado = RoundRectangle2D.Double(x, y, ancho, alto, ancho_Arco, alto_Arco)
  - Arc2D arco = Arc2D.Double(x, y, ancho, alto, ángulo_Inicial, ángulo_Arco, cierre del arco)

    Donde el cierre del arco puede ser:
    - Arc2D.PIE: el arco se cierra con dos líneas que van de punto inicial del arco al centro del arco y del centro del arco al punto final del arco
    - Arc2D.CHORD: el arco se cierra con una línea que va del punto inicial del arco al punto final
    - Arc2D.OPEN: el arco se queda abierto.
  - Line2D linea = Line2D.Double(x_Inicial, y_Inicial, x_Final, y_Final)
  - QuadCurve2D curvaCuadratica = QuadCurve2D.Double(x_Inicial, y_Inicial, x_punto_de_control, y_punto_de_control, x_Final, y_Final)
  - CubicCurve2D curvaCubica = CubicCurve2D.Double(x_Inicial, y_Inicial, x_punto_de_control_1, y_punto_de_control_1, x_punto_de_control_2, y_punto_de_control_2, x_Final, y_Final)
  - Point2D punto = Point2D.Double(x,y)
  - Polygon(int valoresDeX[],int valoresDeY[], int numeroDePuntos)

Donde x, y son respectivamente la x y la y de la esquina superior izquierda de la figura que se dibuja

Además de los constructoras de formas anteriores existe una versión con *Float*. Es mejor usar la versión con *Double* para evitar el uso de moldes, a no ser que se haya de usar gran cantidad de formas geométricas. En este último caso conviene el uso de Float para ahorrar memoria.

Si se elige la opción de constructor referente a *Double* todos los argumentos serán *double* mientras que si la opción seleccionada es *Float* los argumentos serán *float*.

Otros métodos interesantes a la hora de dibujar una figura geométrica son los accesores y mutadores de cada forma.

Texto y Fuentes

Cuando se quiere elegir el tipo de fuente que se usará para nuestras aplicaciones hay dos opciones:

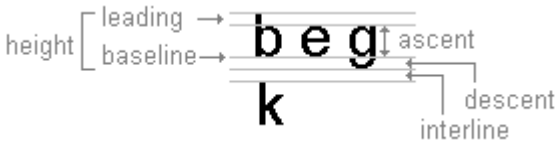
- Usar las fuentes disponibles de AWT, las cuales son: SansSerif, Serif, Monospaced, Dialog y DialogImput
- Usar fuentes de fuera de Java:

```
URL url=new URL("...")
ImputStream in=url.openStream();
Font f=Font.createFont(Font.TRUETYPE_FONT,in);
```

Para consultar las fuentes disponibles en la computadora usaremos:

```
String [] fontNames = GraphicsEnvironment. getLocalGraphicsEnvironment(). getAvailableFontFamilyNames();
```

Cuando se escribe un texto utilizando la clase *Font* hay que tener en cuenta las medidas de las letras. Cada letra tiene su propia altura (height) y anchura(width). Las letras se escriben a partir de una línea de referencia (baseline). La altura está formada por el descent, ascent y leading.



Métodos:

- Clase *Font*:
  - Font(String nombreFuente, int estilo, int tamaño)

  Donde el estilo puede ser:
    - Font.BOLD
    - Font.ITALIC
    - Font.PLAIN
    - Font.BOLD + Font.ITALIC
      - String getFontName()
      - String getFamily()
      - String getName()
      - Rectangle2D getStringBounds(String cadena, FontRenderContext contexto)
      - LineMetrics getLineMetrics(String cadena, FontRenderContext contexto)
      - Font deriveFont(int estilo)
      - Font deriveFont(float tamaño)
      - Font deriveFont(int estilo, float tamaño)
- Clase *LineMetrics*

- float getAscent()
- float getDescent()
- float getLeading()
- float getHeight()

- Clase *Graphics*
  - void setFont(Font fuente)
  - void drawString(String cadena, int x,int y)
- Clase *Graphics2D*
  - void drawString(String cadena, float x, float y)
  - FontRenderContext getFontRenderContext()

Relleno

Para rellenar un objeto usaremos *setPaint*. Este método puede recibir:

- Un Gradiente: Usaremos para ello el constructor de la clase *GradientPaint*

  GradientPaint(Pont p1,Color c1,Pont p2,Color c2)
- Un color:

  Clase *Color*
  - public Color(int rojo, int verde, int azul)
  - public Color(float rojo, float verde, float azul)
  - public void setColor (Color c)
- Una textura *TexturePaint*.

  Para dar una explicación más ilustrativa se usará un ejemplo de creación de una textura
  1. Se crea un patrón de relleno de tamaño 11x11 en color

     BufferedImage bufferedImage = new BufferedImage(11, 11, BufferedImage. TYPE_INT_ARGB);
  2. Se crea un objeto graphics para pintar dentro del BufferedImage

     Graphics gg = bufferedImage.createGraphics();
  3. Se establece el dibujo de la textura, en este caso:

     ```
     gg.setColor(Color.gray);
     gg.fill3DRect(0, 0, 10, 10, true);
     gg.setColor(Color.blue);
     gg.drawRect(1, 1, 3, 3);
     Rectangle2D anchor = new Rectangle2D.Double(0, 0,
                                  bufferedImage.getWidth(),
                                  bufferedImage.getHeight());
     ```
  4. Se crea la textura de tamaño anchor y se rellena con el dibujo creado con anterioridad

     ```
     Paint paint = new TexturePaint(bufferedImage, anchor);
     g2.setPaint(paint);
     ```

Trazo

Para establecer un nuevo trazo usaremos el método setStroke(Stroke stroke) de la clase *Graphics*.

- Clase *Stroke*:
  - BasicStroke(ancho en pixels)
  - BasicStroke(float anchoLinea,int FormaFinaldeLinea, int formaUniondeLineas);
  - BasicStroke(float anchoLinea,int FormaFinaldeLinea, int FormaUniondeLineas, float anguloLimite)
    Donde ángulo límite es el ángulo por debajo del cual una unión inglete es biselada
  - BasicStroke(float anchoLinea, int FormaFinaldeLinea, int FormaUniondeLineas, float anguloLimite, float array de longitud de segmentos[] ,float dashFase)

La forma del final de la línea puede ser:

La forma de la unión de líneas puede ser:

Formas complejas

Para crear formas complejas, es decir, aquellas que se forman por composición de formas simples(rectángulos, líneas, elipses, arcos) se usará *GeneralPath*. Veamos un ejemplo:

Se crea el GeneralPath que usaremos como contenedor de formas:

GeneralPath cjtoFormas=GeneralPath();

Nos situamos en el punto inicial de nuestro conjunto de formas

cjtoFormas.moveTo(10,10);

Se añade al GeneralPath una línea que va desde el punto anterior, en este caso el punto (10,10) al punto (10,20)

cjtoFormas.LineTo(10,20);

Se añade al path una forma geométrica.

En este caso no deseamos que se conecte el último punto de la forma anterior con el primer punto de la forma que se añade, para ello ponemos el segundo argumento del método appent a falso.

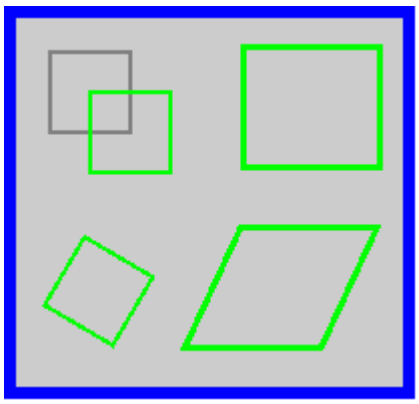
```
Rectangle2D r=....;
cjtoFormas.append(r,false);
```

Se cierra el GeneralPath

```
cjtoFormas.close()
```

Transformaciones de coordenadas

Veamos más claramente la forma de uso realizando el siguiente ejemplo:



Se realiza un movimiento pero sin mover el eje de coordenadas, el cual sigue siendo el (0,0)

```
g2.translate(110, 80);
Rectangle2D square = new Rectangle2D.Double(5, 5, 40, 40);
```

Se dibuja con respecto al punto donde nos trasladamos

```
g2.draw(square);
AffineTransform t = new AffineTransform();
```

Se indica la transformación del eje de coordenadas a realizar (en este caso traslación). No se tiene en cuenta las transformaciones anteriores

```
t.setToTranslation(250, 100);
g2.setTransform(t);
g2.draw(square);
```

Se indica lo Transformación del eje de coordenadas a realizar en este caso rotación

```
t.setToRotation(Math.toRadians(30));
```

Traslado sin trasladar el eje de coordenadas

```
g2.translate(0, 70);
```

Se hace la rotación con respecto a la traslación anterior. Concatenamos transformaciones

```
g2.transform(t);
g2.draw(square);
```

Sin tener en cuenta las transformaciones anteriores se escala

```
t.setToScale(1.7, 1.5);
g2.setTransform(t);
g2.translate(190, 50);
g2.draw(square);
```

Se realiza una transformación de distorsión con respecto a la traslación anterior. Se combinan transformaciones

```
t.setToShear( -0.4, 0);
g2.transform(t);
g2.translate(25, 60);
g2.draw(square);
```

Además de las capacidades descritas con anteriordad existen otras muchas capacidades a la hora de dibujar en Java tales como filtrado de imágenes, manipulación de imágenes, hints de representación, transparencia y composición, recortes, definición de formas mediante la composición de áreas.

Apéndice B: Hebras

Los programas multihebra son programas que dan la impresión de realizar varias tareas a la vez, cada una de las cuáles serán conducidas por un hebra o hilo.

La diferencia entre procesos múltiples y hebra es que cada proceso posee su conjunto de variables mientras que los hebras comparten los mismos datos. Además la comunicación entre hebras es más rápida y menos restrictiva que la de los procesos.

Para que una clase use hebras debe se subclase de *Thread*, pero si ya fuera subclase de otra la clase debería implementar *runnable*, ya que Java no permite la herencia múltiple.

Métodos:

- *Hebra()* construye un hebra para iniciarlo es necesario llamar al método *start()*
- *void start()* inicia el hilo. Este método llama automáticamente a *run*.

- *static boolean interrupted().* Comprueba si el hebra actual ha sido interrumpido. Reinicia el estado interrumpido del hebra actual a false.
- *boolean isInterrupted()* Comprueba si una hebra concreta ha sido interrumpido. *boolean isAlive()*. Para determinar si un hebra está vivo
- *void sleep*(long milisegundos)
- *void wait*(long milisegundos)
- *void join*() espera a que el hebra especificado deje de estar activo.

Cuando un hebra está durmiendo no se puede comprobar activamente si se debe finalizar.

La excepción i*nterruptedException* la disparan los métodos *wait* y *sleep* cuando hebra se detiene debido a que otro ha llamado a *interrupt*. Un hebra se interrumpe cuando termina.

BIBLIOGRAFÍA

1. Java : how to program. Autores: H.M. Deitel, P.J. Deitel. Publicación: Upper Saddle River (New Jersey): Prentice Hall, cop. 2002. 4ª edición.

2. Advanced Java 2 Platform : how to Program. Autores: H. M. Deitel, P. J. Deitel, S. E. Santry. Publicación: Upper Saddle River (New Jersey): Prentice Hall, cop. 2002. Colección: How to program series.

3. Java 2. Vol. 1, Fundamentos. Autores: Cay S. Horstmann, Gary Cornell. Traducción, KME Sistemas, S.L. Publicación Madrid: Prentice-Hall, D.L. 2003

4. Java 2. Vol. 2, Características avanzadas. Autores: Cay S. Horstmann, Gary Cornell . Traducción, KME Sistemas, S.L. Publicación: Madrid Prentice-Hall, D.L. 2002

5. Piensa en Java. Autor: Bruce Eckel. Traducción de Jorge González Barturen. Título original: Thinking in Java. 2nd. edición. Revisión técnica de Javier Parra Fuente y Ricardo Lozano Quesada. Coordinación general y revisión técnica de Luis Joyanes Aguilar. Publicación: Madrid: Prentice Hall, 2002

6. Fundamentos de algoritmia. Autores: G. Brassard, P. Bratley. Traducción de Rafael García-Bermejo. Revisión técnica de Narciso Martí, Ricardo Peña y Luis Joyanes Aguilar. Publicación: Madrid: Prentice Hall, 2002. 1ª edición en español.

7. Diseño de programas : formalismo y abstracción. Autor: Ricardo Peña Marí. Publicación México : Prentice Hall, D. L. 1993

8. Data Structures and Algorithm Analysis in Java. Autor Mark Allen Weiss, Florida International University. Publicación: Addison-Wesley, 1999

9. Estructuras de datos y métodos algorítmicos : Ejercicios resueltos. Autores: Narciso Martí Oliet, Yolanda Ortega Mallén, José Alberto Verdejo López. Publicación: Madrid : Pearson Prentice-Hall, 2003. Colección Prentice Práctica.

10. Apuntes del profesor Jose Luis Sierra de la UCM de la asignatura de Laboratorio de Programación 3

Páginas Web

- Estructuras de datos y algoritmos

11. http://www.nist.gov/dads/#H
12. http://www.itlp.edu.mx/publica/tutoriales/estru1/
13. http://www.dcc.uchile.cl/~cc30a/apuntes/Estructuras/
14. http://mailweb.pue.udlap.mx/~ccastane/Syllabus_Estructura_Datos/Sy_EstructuraDatos_Java.html

- Árboles
  15. http://www.hci.uniovi.es/martinDocencia/DSTool/AvlTreePage.htm
  16. http://c.conclase.net/edd/index.php
  17. http://www.dei.inf.uc3m.es/docencia/p_s_ciclo/edii/apuntes/tema5-parte1.pps
  18. http://www.lcc.uma.es/~galvez/ftp/tad/tadtema4cont.pdf

- Colas de Prioridad
  19. http://ciips.ee.uwa.edu.au/~morris/Year2/PLDS210/heapsort.html

- Grafos
  20. http://www.cs.buap.mx/~titab/files/grafos1.pdf
  21. http://www.cs.buap.mx/~titab/files/grafos2.pdf
  22. http://www.csse.monash.edu.au/~lloyd/tildeAlgDS/Graph/Directed/ ->
  23. http://www.csse.monash.edu.au/~lloyd/tildeAlgDS/Graph/Undirected/ ->
  24. http://www.dma.fi.upm.es/gregorio/grafos/paginagrafos.html

- Tutoriales Java

  25. http://www.itapizaco.edu.mx/paginas/JavaTut/froufe/introduccion/indice1.html

PÁGINA DE AUTORIZACIÓN

Se autoriza a la Universidad Complutense a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la propia memoria, como el código, la documentación y/o el prototipo desarrollado.

Laura Gutiérrez García Esther Rico Redondo Carmen Torrano Giménez